

**Politechnika Wrocławskaw
Wydział Informatyki i Telekomunikacji**

Kierunek: **Informatyka Techniczna (ITE)**
Specjalność: **Inżynieria Systemów Informatycznych (INS)**

**PRACA DYPLOMOWA
MAGISTERSKA**

**Wykorzystanie algorytmów genetycznych
w systemach wykrywania intruzów w sieciach
komputerowych**

inż. Bartosz Błyszcz

Opiekun pracy
dr inż. Tomasz Babczyński

Słowa kluczowe: 3-6 słów

WROCŁAW, 2024

Streszczenie

Wykaz skrótów

Tabela 1. Tabela skrótów
Źródło: opracowanie własne

GA	<i>Genetic Algorithm</i>	Algorytm Genetyczny
GP	<i>Genetic Programming</i>	Programowanie Genetyczne
GNB	<i>Gaussian Naive Bayes</i>	Naiwny Klasyfikator Bayesa wykorzystujący rozkład Gaussa
ANN	<i>Artificial Neural Network</i>	Sztuczna sieć neuronowa
CNN	<i>Convolutional Neural Network</i>	Konwolucyjna sieć neuronowa
ML	<i>Machine Learning</i>	Uczenie maszynowe
AI	<i>Artificial Intelligence</i>	Sztuczna Inteligencja
IDS	<i>Intrusion Detection System</i>	System Wykrywania Intruzów
SVM	<i>Support Vector Machine</i>	Maszyna Wektorów Nośnych
AUC	<i>Area Under Roc Curve</i>	Przestrzeń pod krzywą ROC
LCDPs	<i>Low-code Development Platforms</i>	Platforma Low-code
BI	<i>Business Intelligence</i>	Narzędzia biznesowe do przekształcania danych
CDN	<i>Content Delivery Network</i>	Sieć dostarczania zawartości
MLP	<i>Multilayer Perceptron network</i>	Sieć wielowarstwowa perceptronowa
Azure ML	<i>Azure Machine Learning Studio</i>	
GAGNB	<i>Gaussian Naive Base - with GA</i>	
HTTPS	<i>Hyper Text Transfer Protocol Secure</i>	Protokół służący do komunikacji w sieci internetowej
REST	<i>Representative State Transfer</i>	Rozwiązanie architektoniczne służące do komunikacji w sieci internetowej

Spis treści

1. Wstęp.....	7
1.1. Wprowadzenie i uzasadnienie tematu pracy.....	7
1.2. Cel pracy dyplomowej.....	7
1.3. Założenie techniczne	8
2. Sztuczna inteligencja	9
2.1. Uczenie maszynowe	9
2.1.1. Uczenie nadzorowane	10
2.1.2. Uczenie nienadzorowane	11
2.1.3. Uczenie przez wzmacnianie	11
2.1.4. Uczenie częściowo nadzorowane	12
2.2. Sieć neuronowa.....	12
2.2.1. Głębokie uczenie.....	15
3. Klasyfikacja danych	17
3.1. Metryki	17
3.1.1. Dokładność	18
3.1.2. Precyzyja.....	18
3.1.3. Czułość.....	18
3.1.4. F1	18
3.1.5. AUC	18
4. Podejście low-code/no-code	19
4.1. Platformy	20
4.1.1. Microsoft PowerApps	20
4.1.2. Amazon QuickSight.....	20
4.1.3. Google AppSheet.....	21
5. Microsoft Azure.....	23
5.1. Infrastruktura	24
5.2. Machine Learning Studio	25
6. Opis doświadczenia.....	27
6.1. Dane	27
6.2. Środowisko programistyczne.....	28
6.3. Algorytmy	30
6.3.1. Two-Class Support Vector Machine	30
6.3.2. Two-Class Boosted Decision Tree.....	31
6.3.3. Two-Class Decision Forest	31

6.3.4.	Two-class Neural Network	32
6.3.5.	Two-Class Average Perceptron.....	33
6.3.6.	Autorskie rozwiązanie	34
6.3.7.	DANet.....	35
7.	Przebieg badań	37
7.1.	Metodologia badawcza	37
7.2.	Przygotowanie platformy.....	38
7.2.1.	Przygotowanie danych.....	38
7.2.2.	Trenowanie oraz testowanie algorytmów	38
7.2.3.	Utworzenie tabeli porównawczej.....	40
7.3.	Weryfikacja potoku.....	41
7.4.	Próba badawcza	43
7.4.1.	Dopasowanie.....	45
7.4.2.	Precyzja.....	46
7.4.3.	Czułość.....	47
7.4.4.	F1	48
7.4.5.	AUC	49
7.5.	Analiza wyników	50
7.6.	Wnioski.....	50
8.	Perspektywy rozwoju.....	53

1. Wstęp

1.1. Wprowadzenie i uzasadnienie tematu pracy

Klasyfikacja danych tabelarycznych jest zagadnieniem, które na codzień dostarcza wyzwań jej twórcom z powodu mnogości danych, a także mnogości cech, a także z nierzadko małą ilością próbek. Jednym z problemów jest między innymi wybór odpowiedniego algorytmu do problemu. Dane tabelaryczne występują w każdej dziedzinie, przez co raz na jakiś czas proponowane są nowe rozwiązania i algorytmy mające rozwiązać problem klasyfikacji w sposób lepszy i wydajny. Część twórców próbuje podchodzić do tego w sposób innowacyjny, lecz nie zawsze to wychodzi z powodu chociażby dosotowania algorytmu pod konkretną strukturę danych, co powoduje problemy z wykorzystaniem rozwiązania dla innych danych.

Obecnie jednymi z najpopularniejszych algorytmów do klasyfikacji danych są logiczna regresja(*ang. logistic regression*), drzewo decyzyjne(*ang. decision tree*), losowy las(*ang. random forest*), maszyna wektorów nośnych(*ang. support vector machine*), naiwny bayes(*ang. Naive Bayes*). Dlatego też bardzo ważne jest porównanie wytworzzonego wcześniej rozwiązania z grupą innych algorytmów, które próbują przetworzyć ten sam zestaw danych.

W dzisiejszych czasach taka próba jest bardzo uproszczona chociażby przez takie platformy jak *Machine Learning Studio*, które pozwalają na wykorzystanie mocy obliczeniowej sklasteryzowanych jednostek wirtualnych do wykonywania obliczeń na odpowiednich maszynach wirtualnych, a także do budowania skomplikowanych zautomatyzowanych procesów złożonych z wielu zadań(*ang. pipeline*). W związku z czym możliwość wykorzystania platformy chmurowej pozwoli na zautomatyzowanie procesu porównawczego oraz oddelegowanie zadań od chmury obliczeniowej co pozwoli na uniezależnienie powodzenia doświadczenia od mocy obliczeniowej komputera lokalnego, a także na ukazanie całościowo procesu porównania algorytmów klasyfikacyjnych.

1.2. Cel pracy dyplomowej

Celem niniejszej pracy dyplomowej jest porównanie algorytmu klasyfikacji danych tabelarycznych wypracowanego w trakcie pisania pracy inżynierskiej, do algorytmów dostępnych w aplikacji *Machine Learning Studio* znajdującej się na platformie *Microsoft Azure*.

1.3. Założenie techniczne

Dane prezentowane w **tabeli 1.1** określają podstawowe założenia techniczne przyjęte w trakcie wykonywania analizy porównawczej. Dane te dotyczą między innymi środowiska, w którym wykonane było doświadczenie. Dodatkowo uwzględniono zestaw danych oraz biblioteki użyte w trakcie tworzenia doświadczenia.

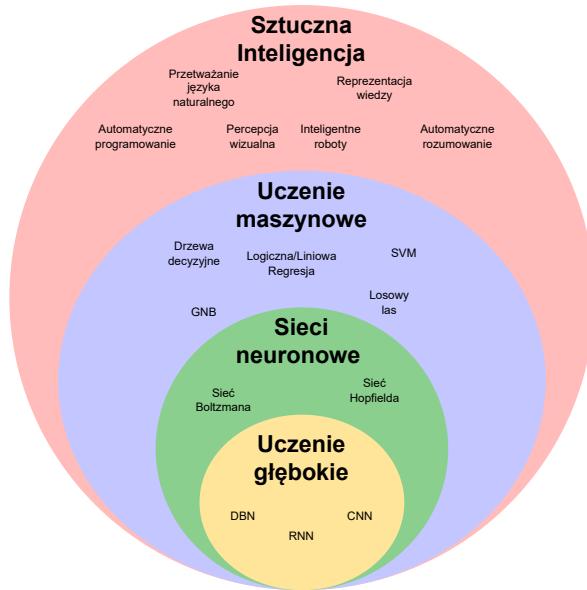
Tabela 1.1. Założenia techniczne pracy dyplomowej
Źródło: Opracowanie własne

Środowisko uruchomieniowe	Machine Learning Studio[1]
Język oprogramowania	Python 3.x
Wykorzystane biblioteki	scikit-learn [scikit-learn]
	Numpy [Harris2019]
	Pandas [2, 3]
Wykorzystane dane	CICDS2017 [4]

2. Sztuczna inteligencja

Według słownika *Oxford English Dictionary* słowo **"inteligencja"** oznacza zdolność do rozumienia, a analizy i dostosowania się do zmian[5].

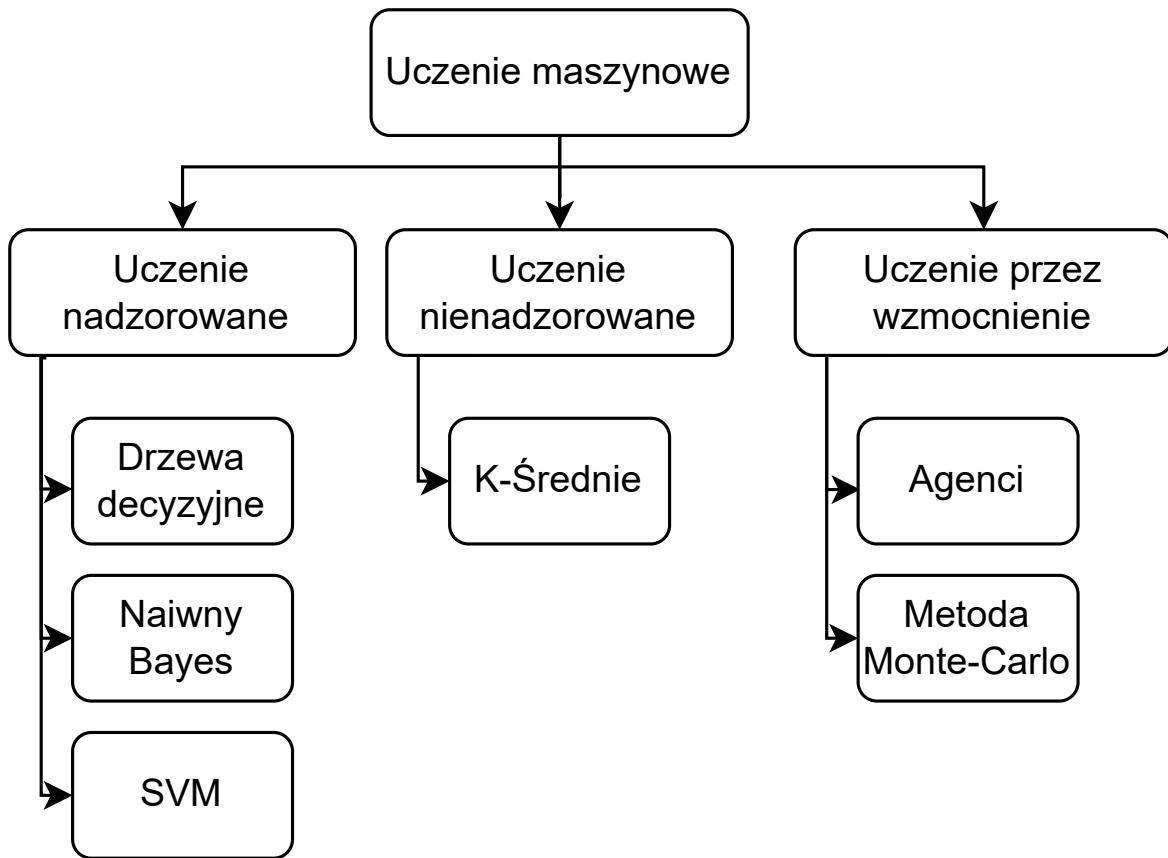
Sztuczna inteligencja(*ang. Artificial Intelligence*) (**AI**) jest wykorzystywana na wiele sposobów podczas prowadzenia badań naukowych: od stawiania hipotez oraz budowania twierdzeń matematycznych, tworzenia i monitorowania badań, zbierania danych i wielu innych czynności towarzyszącymi podczas badań. Najpopularniejszymi zastosowaniami jest między innymi uczenie nienadzorowane oraz wykrywanie anomalii[6, 7]. Schemat podziału sztucznej inteligencji pokazano na **obrazie 2.1**.



Rys. 2.1. Graficzne przedstawienie podziałów sztucznej inteligencji
 Źródło: [8]

2.1. Uczenie maszynowe

Uczenie maszynowe(*ang. Machine Learning*) (**ML**) jest to dziedzina nauki nad algorytmami oraz modelami statystycznymi, które mogą być wykorzystywane do specyficznych zadań na przykład klasyfikacji, rozpoznawania obrazów bądź mowy, a dodatkowo nie są zaprogramowane specyficznie pod konkretne zadanie, a jedynie pod grupę zadań tak jak pokazano na **obrazie 2.2**. Dlatego też nie ma jednego najlepszego rozwiązania, które można wykorzystać w każdym przypadku. Wykorzystanie konkretnego algorytmu determinuje typ zadania jaki ma być rozwiązany.

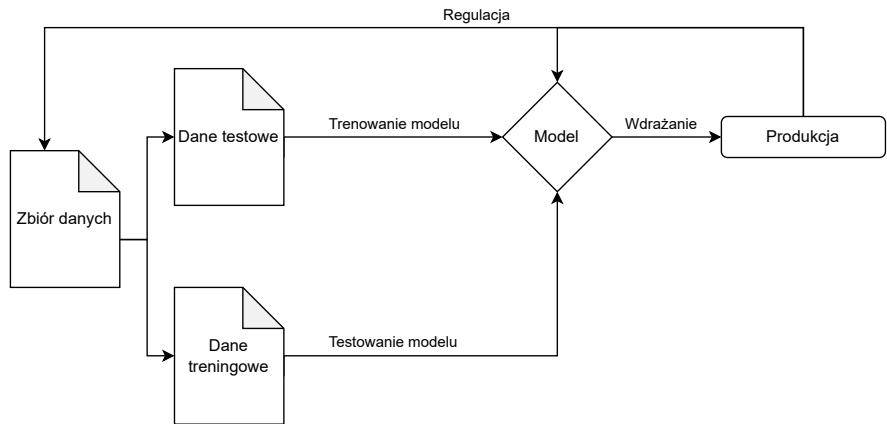


Rys. 2.2. Podział uczenia maszynowego

Źródło: [7]

2.1.1. Uczenie nadzorowane

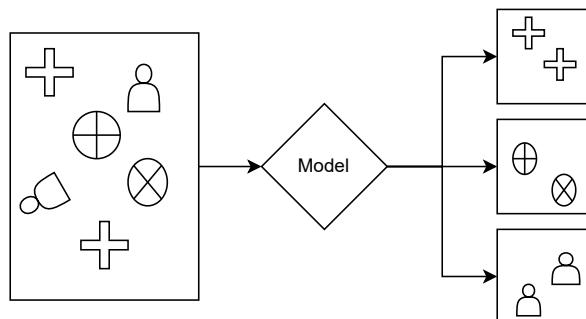
- w trakcie tego uczenia stosuje się zbiór posiadający etykiety. Model uczy się przyporządkowywać określone cechy do konkretnych kategorii. Dane wejściowe dzielone są na dane treningowe i dane testowe. Zbiór treningowy jest wykorzystywany do trenowania modelu, a zbiór testowy do sprawdzenia rezultatu, na bazie którego może nastąpić korekta uczenia zilustrowano to **obrazem 2.3**. Algorytmy uczenia nadzorowanego można zastosować między innymi do weryfikacji ruchu sieciowego w celu określenia czy ruch bezpieczny, przez co można to zastosować w systemach wykrywania intruzów(*ang. Intrusion Detection System*) (**IDS**). Algorytmy wchodzące w skład uczenia nadzorowanego to między innymi klasyfikacja naiwna bayesa, drzewa decyzyjne, maszyny wektorów wektorów nośnych[6, 7].



Rys. 2.3. Uczenie nadzorowane
 Źródło: [7]

2.1.2. Uczenie nienadzorowane

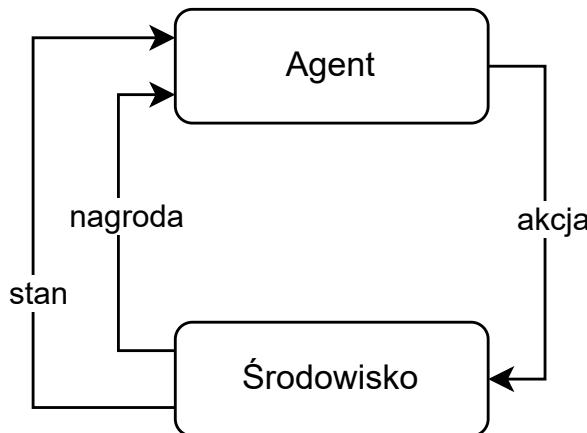
- w tym przypadku nie wykorzystuje się zbioru oznaczonego, algorytm sam próbuje odkryć prawidłową odpowiedź. Dzieje się tak, w danych, których nie da się nazwać albo doprecyzować. Wykorzystuje się to do między innymi detekcji anomalii, co pozwoli do na przykład wykrycia zbyt dużego zużycia prądu w pokoju domu studenckiego dzięki czemu uda się wyłapać nieautoryzowaną koparkę kryptowalut. Dodatkowo można wykorzystać je do szukania wzorców, albo zarządzania magazynem. W skład takich algorytmów wchodzi: K-średnie, klasteryzacja. Schemat uczenia nienadzorowanego poprzez klasteryzację jest pokazany na **obrazie 2.4**.



Rys. 2.4. Uczenie nienadzorowane
 Źródło: Opracowanie własne

2.1.3. Uczenie przez wzmacnianie

- jest to uczenie poprzez nagradzanie dobrych rozwiązań, a karanie złych, potocznie mówiąc jest to metoda *"kija i marchewki"*. Wykorzystywana w trenowaniu pojazdów autonomicznych pozwala na nagradzanie pojazdów za wybór lepszych tras przykładowo za wybór dróg asfaltowych zamiast polnych. Skupia się w dużym stopniu na agencie i jego decyzyjach w danym środowisku co pokazano na **schemacie 2.5**. Należy do jednych z trzech głównych paradymatów obok uczenia nadzorowanego i nienadzorowanego.



Rys. 2.5. Uczenie przez wzmacnianie
Źródło: [7]

2.1.4. Uczenie częściowo nadzorowane

- do trenowania takich modeli stosuje się niewielkie zbiory oznaczone, oraz większe zbiory nieoznaczone, dzięki którym można próbować rozpoznać rozległe zbiory danych na podstawie pewnych cech wspólnych. Stosuje się to ze względu na mnogość danych na świecie, których opisanie byłoby niemożliwe oraz albo zbyt kosztowne. Przykładowo można znaleźć zastosowanie tych algorytmów w bankowości albo klasyfikowaniu stron internetowych poprzez wyszukiwanie treści na stronie i kategoryzowaniu ich[9]. Jest to połączenie uczenia nadzorowanego i nienadzorowanego[7].

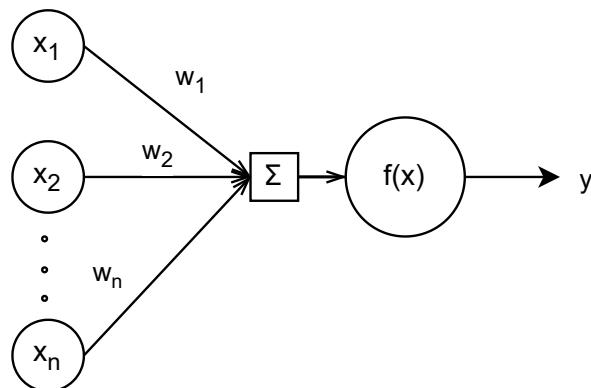
2.2. Sieć neuronowa

Ludzki mózg jest najbardziej złożonym organem znany ludziom. Badacze zainspirowani jego strukturą składającą się z połączonych ze sobą komórek neuronowych, które przetwarzają równolegle wiele informacji, próbują przenieść pewien poziom inteligencji do komputerów. Przykładem tego jest wiele algorytmów, wchodzących w skład sztucznych sieci neuronowych(*ang. artificial neural network*) (ANN), między innymi sieci Kohonena, sieci Hopfielda, sieci konwolucyjne. Sieci te próbują w pewien sposób odwzorować próbę na przykład klasyfikacji danych przez jednostkę wzorowaną na ludzkim mózgu, pomimo tych osiągnięć symulacja ludzkiej świadomości oraz emocji wciąż jest jedynie w sferach fantazji naukowych[Wang2003].

Sieć neuronowa jest zbudowana z połączonych ze sobą warstw neuronów tak jak na **rysunku 2.7**, które w pewien sposób mają wykonać zadania uczenia maszynowego. Najprostszym przykładem sieci neuronowej jest pojedynczy neuron, który może służyć do prostych zadań klasyfikacyjnych: **schemat 2.6**. Sieć ta potrafi się dostosowywać do danych wejściowych tak aby uzyskać odpowiedni wynik, wykonuje w tedy proces uczenia stosując do tego na przykład algorytm wstępnej propagacji wag. W zależności od problemu istnieje wiele różnych sieci, które można zastosować. Jednym z trudniejszych rzeczy w doborze sieci jest dobór warstw ukrytych oraz ilości neuronów, ponieważ w tym celu twórcy mogą opierać się jedynie na własnej wiedzy i doświadczeniu. Podstawy teorii sieci neuronowych zostały stworzone w połowie XX wieku. Złota era uczenia maszynowego rozpoczęła

się dopiero na początku XXI wieku, kiedy to jednocześnie pojawiły się takie trendy jak: Big Data, redukcja kosztów obliczeń równoległych, oraz pierwsze badania nad głębokimi sieciami neuronowymi(*ang. Deep Neural Network*) (**DNN**). Największe zastosowanie DNN miało miejsce dopiero w ostatniej dekadzie kiedy to pojawiły się:

- **Google Braine** - grupa badawcza założona w 2011 roku, zajmująca się badaniami nad sztuczną inteligencją
- **DeepFace** - rozwiązanie stworzone przez firmę Facebook w 2014 roku, służące do rozpoznawania twarzy na zdjęciu [10, 11].



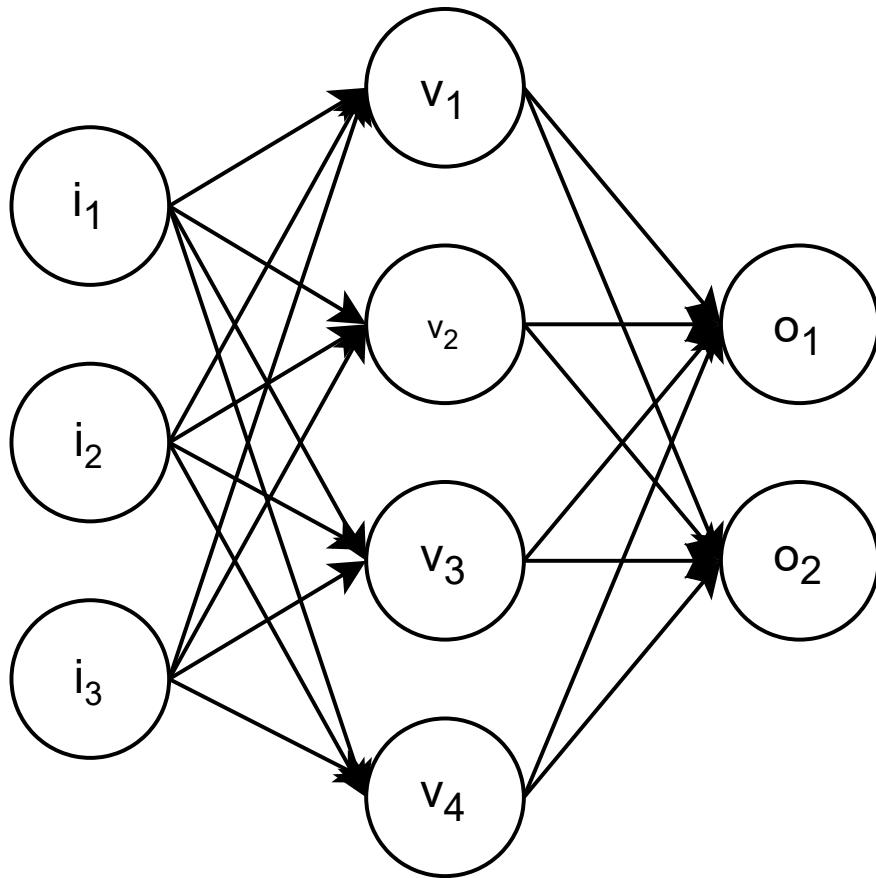
Σ : sumator

$f(x)$: funkcja aktywacyjna

$w_x \forall x \in [1, 2, \dots, n]$: wagi

$x_x \forall x \in [1, 2, \dots, n]$: wejścia

Rys. 2.6. Schemat neuronu
Źródło: Opracowanie własne



$i_x \forall x \in [1, 2, 3]$: dane wejściowe
 $v_x \forall x \in [1, 2, 3, 4]$: neurony w warstwie ukrytej
 $o_x \forall x \in [1, 2]$: dane wyjściowe

Rys. 2.7. Schemat sieci neuronowej
 Źródło: Opracowanie własne

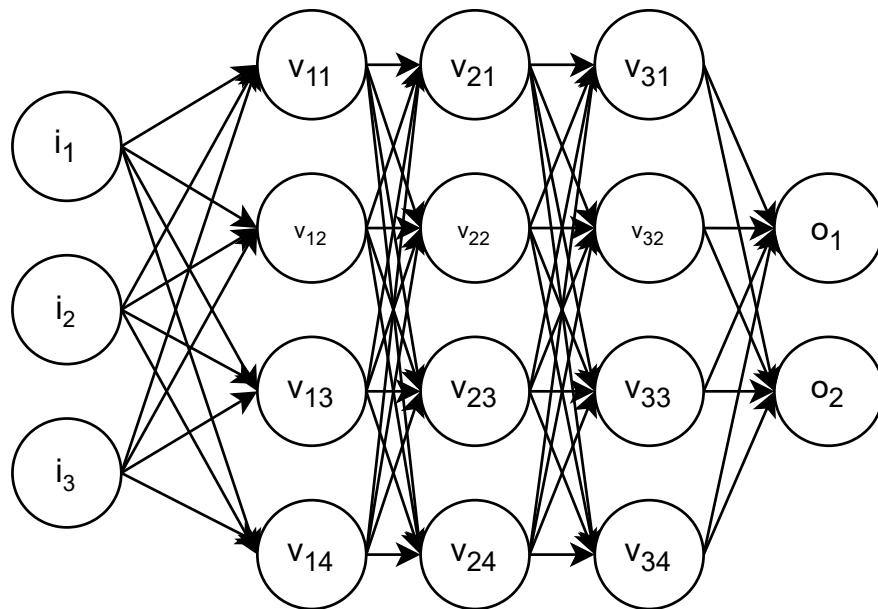
Sieci neuronowe możemy podzielić na wiele rodzajów, do których możemy zaliczyć między innymi:

- **perceptron** - jest to najstarszy przykład sieci neuronowej złożonej z jednego perceptronu (neuronu). Można je zastosować w problemach klasyfikacji;
- **sieci wielowarstwowe perceptronowe** - jest to sieć złożona z wielu warstw połączonych ze sobą neuronów, najprostszy model sieci zaprezentowany na **rysunku 2.7**. Składa się z warstwy wejściowej, warstw (jednej bądź wielu) ukrytych oraz warstwy wyjściowej. W neurony w tej sieci w porównaniu do perceptronów, mają funkcję aktywacyjną sigmoidalną, ze względu na rozwiązywanie problemów nieliniiowych (posiadających więcej rozwiązań niż dwa 0/1). Można je zastosować na przykład do klasyfikacji danych;
- **sieci konwolucyjne (CNN)** - są to sieci służące do rozpoznawania obrazów, nazwa wzięła się od wykonywanej na obrazie operacji konwolucji (splotu). Sieci te posiadają dodatkowe warstwy konwolucyjne oraz spłaszczenia, które pozwalają zamienić reprezentację obrazu w pojedyncz;

- **sieci rekurencyjne** - charakteryzują się pętlą zwrotną w warstwie ukrytej. Mogą być wykorzystane do generowania tekstu, tłumaczeń maszynowych, a także na przykład przewidywania cen rynkowych;
- **sieci samoorganizujące się** - wykorzystuje uczenie nienadzorowane oraz. Składają się jedynie z warstwy wejściowej i wyjściowej. Zaś cechą charakterystyczną jest to, że neurony określające podobne klasy znajdują się obok siebie. Sieci te mogą być wykorzystywane do podziału klientów na odpowiednie grupy bądź do wskazania jakim klientom zaproponować karty kredytowe [12, 13].

2.2.1. Głębokie uczenie

Jest to podkategoria uczenia maszynowego polegająca na tworzeniu wielowarstwowych sieci neuronowych. W porównaniu do podstawowych sieci neuronowych potrzebuje ogromnych zbiorów danych do utworzenia modelu predykcyjnego. Potrzebuje również dużo więcej mocy obliczeniowej przez wzgląd na ilość warstw ukrytych, których może być dużo więcej, przykładem najprostszej sieci głębokiej jest **obraz 2.8**.



$i_x \forall x \in [1, 2, 3]$: dane wejściowe

$v_x \forall x \in [11, 12, 13, 21, 22, 23, 31, 32, 33]$: neurony w warstwie ukrytej

$o_x \forall x \in [1, 2]$: dane wyjściowe

Rys. 2.8. Schemat prostej głębokiej sieci neuronowej
 Źródło: Opracowanie własne

Warstwa wyjściowa DNN może dostarczać dane różnego formatu, może to być na przykład, tekst, liczba bądź dźwięk. Posiada również bardzo dużo zastosowań w których skład wchodzi generowanie treści, Deepfake, analiza obrazów, wskazywanie obiektów na obrazach, projektowanie leków, czatboty. Jest to udoskonalenie podstawowych sieci neuronowych. Tak więc część typów sieci opisanych w sekcji 2.2 będzie odnosić się do głębokich sieci neuronowych, należy do nich CNN[MicrosoftDepp2023].

3. Klasyfikacja danych

Klasyfikacja jest to próba rozpoznania obiektów na bazie ich cech. Jest to jedna z pierwszych rzeczy jaką uczą się niemowlęta, zaczynając od rozpoznania rodziców, próby rozróżnienia kształtów, kolorów, rzeczy. W otaczającym świecie istnieje wiele mechanizmów mających sklasyfikować rzeczy. Należą do nich katalogi biblioteczne, klasyfikacja trunków, kaw, pojazdów, produktów spożywczych i wielu innych rzeczy. Człowiek od zawsze próbuje skategoryzować i uporządkować posiadaną wiedzę w zbiory ułatwiające obcowanie z tą wiedzą.

W uczeniu maszynowym klasyfikacja jest to metoda uczenia nadzorowanego, podczas której model próbuje przewidzieć etykietę obiektu na podstawie jego cech. Proces uczenia modelu klasyfikacji jest oparty o dwa zbiory, treningowy i testowy, z czego zbiór treningowy powinien być mniejszy od zbioru testowego. Po udanym procesie trenowania modelu, następuje proces testowania na podstawie którego wylicza się odpowiednie metryki pozwalające na ewaluację modelu. W pracy magisterskiej wykorzystano zbiór danych, który posiada dwa typy etykiet [0, 1], dlatego też na tej podstawie zbudowano macierz pomyłek.

3.1. Metryki

W trakcie ewaluacji algorytmu służącego do klasyfikacji wykorzystuje się metryki bazujące na macierzy pomyłek, która została opisana w **tabeli 3.1**

Tabela 3.1. Macierz pomyłek
 Źródło: Opracowanie własne

		Prawdziwe wartości	
		1	0
Przewidziane wartości	1	TP	FN
	0	FP	TN

- **TP** - prawdziwie pozytywny
- **FN** - fałszywie negatywny
- **FP** - fałszywie pozytywny
- **TN** - prawdziwie negatywny

3.1.1. Dokładność

$$\frac{TP + TN}{TP + TN + FP + FN} \quad (3.1)$$

Stosunek wszystkich dobrze oznaczonych obiektów do liczby wszystkich prób.

3.1.2. Precyzyja

$$\frac{TP}{TP + FP} \quad (3.2)$$

Stosunek poprawnie wybranych obiektów klasy "I", do wszystkich wybranych obiektów tej klasy.

3.1.3. Czułość

$$\frac{TP}{TP + FN} \quad (3.3)$$

Stosunek poprawnie sklasyfikowanych obiektów klasy "I", do wszystkich obiektów, które powinny być w tej klasie.

3.1.4. F1

$$\frac{2 * x * y}{x + y} \quad (3.4)$$

Jest średnia harmoniczna precyzji (x) i czułości (y);

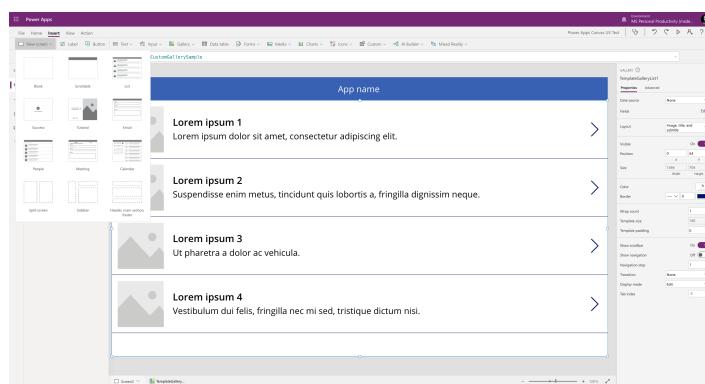
3.1.5. AUC

AUC - (*Area Under Roc Curve*) - Jest to pole pod krzywą ROC pokazuje sprawność klasyfikatora. Im wyższa wartość AUC tym lepiej. Wynik AUC jest z zakresu $<0, 1>$:

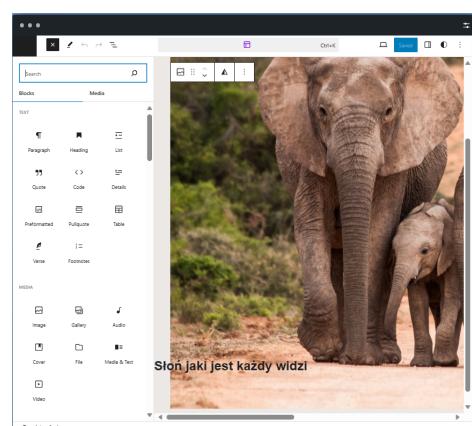
- AUC = 1 - klasyfikator idealny,
- AUC = 0,5 - klasyfikator losowy,
- AUC < 0,5 - klasyfikator gorszy niż losowy[14].

4. Podejście low-code/no-code

Low-Code oraz no-code to nowe podejście skupiające się na umożliwieniu tworzenia programów w sposób nie wymagający znajomości języka oprogramowania. Podejście to ma pozwolić osobom nie będącym programistami na tworzenie aplikacji biznesowych. Ma to zwiększyć tempo tworzenia rozwiązań biznesowych, na które zapotrzebowanie wciąż rośnie. Jednakże podejście to jest stosunkowo świeżym podejściem, którego początki można było obserwować w codziennym życiu na przykład podczas tworzenia stron internetowych korzystając z narzędzi takich jak *Wordpress*, *Joomla*, *Wix*. Narzędzia te umożliwiają w łatwy sposób tworzyć stronę z tak zwanych kafelków, które umieszczone w odpowiednim miejscu były odpowiedzialne za jedną konkretną rzecz[15, 16, 17]. Przykład na **obrazie 4.1, 4.2**.



Rys. 4.1. PowerApps od Microsoft
Źródło: [18]



Rys. 4.2. Wordpress.com
Źródło: [19]

Coraz większa popularnością cieszą się platformy low-code(*ang. Low-code Development Platforms*) (**LCDPs**) dostarczane między innymi przez Google, Microsoft, Amazon pozwalają one na tworzenie wysoko skalowalnych rozwiązań przy niewielkim albo i żadnym nakładzie programowania. Ma to umożliwić osobom z niewielkim doświadczeniem w programowaniu, na szybkie wdrożenie oraz tworzenie niezawodnego oprogramowania. Twórcy platform oferują również korzystającym zmniejszenie ilości pracy potrzebnej do wdrożenia albo rozwijania kolejnych funkcjonalności[20, 21].

4.1. Platformy

LCDP udostępniane twórcom aplikacji, umożliwiają skalowalność rozwiązań tworzonych na własne potrzeby. Dodatkowo są popularne przy tworzeniu aplikacji typu "*aplikacja jako usługa*" (*ang. Software-as-a-Service*) (SaaS), które opłacane są tylko za stopień ich użycia, co w niektórych przypadkach może się okazać dużo bardziej opłacalne niż utrzymywanie swoich rozwiązań serwerowych. Dzięki takim rozwiązaniom wiele małych firm będzie mogło pozwolić sobie na tworzenie i utrzymywanie dostosowanych rozwiązań opartych o ekosystem *Microsoft365 / Google Workspace*.

4.1.1. Microsoft PowerApps

Jest to platforma programistyczna umożliwiająca tworzenie niestandardowych aplikacji dla rozwiązań biznesowych. Umożliwia ona tworzenie aplikacji opartych o różnorakie źródła danych do których należą między innymi: SQL Server, SharePoint, Dynamics 365. Dodatkową zaletą tego rozwiązania jest tworzenie aplikacji responsywnych, działających dobrze na wielu rodzajach urządzeń. Dodatkowo platforma ta pozwala tworzyć trzy typy aplikacji przy braku konieczności kodowania[22]

- **Kanwa** - jest to typ aplikacji oparty o model danych znajdujący się na przykład w Excelu. Aplikację tego typu tworzy się za pomocą przesuwanych kafelek, a proces przypomina po trochę robienie prezentacji przy użyciu aplikacji Powerpoint, co umożliwia pełną dowolność w tworzonym interfejsie graficznym[23].
- **Oparte na modelu** - w ramach korzystania z usługi Microsoft Dataverse można wygenerować aplikacje bazujące na danym modelu danych, przez co użytkownicy otrzymują produkt ułatwiający im analizę danych[24].
- **Karty** - są to uproszczone aplikacje, które można dodać do usługi Microsoft Teams w określonym biznesowym celu. Dużą zaletą tego rozwiązania jest możliwość korzystania z źródeł danych, dzięki czemu poszczególne karty mogą odpowiadać za jedno zadanie biznesowe[25].

4.1.2. Amazon QuickSight

Jest to rozwiązanie firmy Amazon, które umożliwia firmom dostarczanie rozwiązań z zakresu analityki biznesowej(*ang. business intelligence*) (BI) dzięki interaktywnym pulpitom korzystającym z jednego źródła prawdy. Dodatkowo korzystanie z interaktywnych formularzy, raportów oraz zapytań w języku naturalnym pozwala interesariuszom otrzymać możliwość korzystania z jednolitych rozwiązań opartych o różne modele danych[26].

4.1.3. Google AppSheet

Platforma AppSheet od firmy Google umożliwia tworzenie aplikacji mobilnych oraz desktopowych bez użycia kodu. Firma wskazuje na możliwości integracyjne z różnymi dostawcami danych, do których należą między innymi Microsoft, Dropbox, a także wbudowaną integrację z aplikacjami Google Workspace do których należą Gmail, Sheets oraz Spaces. Platforma pozwala również na tworzenie automatycznych botów, które wykonują zadania w oparciu o bodźce zewnętrzne bądź wewnętrzne. Narzędzie pozwala w prosty sposób na tworzenie szybkich rozwiązań biznesowych w oparciu o ekosystem firmy Google[27].

5. Microsoft Azure

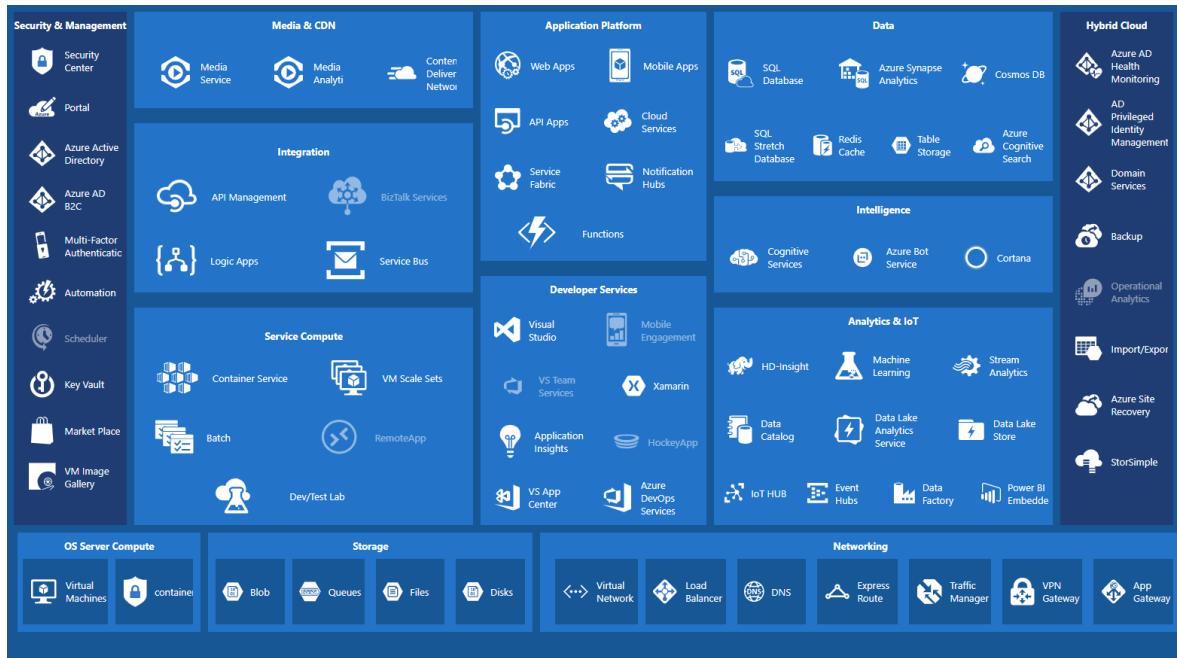
Platforma została oddana do użytku w 2008 roku jako Windows Azure. Usługa ta została zbudowana na modułach Windows NT. Platforma została udostępniona komercyjnie po 2010 roku, kiedy to dodano możliwość korzystania z szerszej ilości usług i języków programowania. Do usług należało między innymi udostępnienie baz danych Microsoft SQL Server opartych o .NET Framework 4; obsługę aplikacji pisanych w języku C#, Java, PHP; sieć dostarczania zawartości(*ang. Content Delivery Network*) (**CDN**).

Następnym krokiem było przemianowanie platformy na Microsoft Azure oraz pójście w kierunku infrastruktury definiowanej jako serwis(*ang. Infrastructure-as-a-Service*) (**IaaS**), oraz powolne adoptowanie usług open-source.

W kolejnej generacji Microsoft zaadoptował rozwiązania Big Data do swojej platformy, umożliwiając korzystanie z języka R, połączenie do Power BI, a także umożliwienie połączenia do rozwiązań end-to-end.

W czwartej generacji platformy, Microsoft skupił się na rozwiązańach uczenia maszynowego oraz integracji z bazami danych, dzięki czemu powstało Azure Machine Learning Studio oraz Azure Machine Learning Operations (MLOps).

Obecnie platforma została wzbogacona o Kuberntesa, dzięki czemu konteneryzacja ułatwia pracę z klastrami wirtualnymi, dzięki którym można w lepszy sposób zarządzać aplikacjami i usługami. Dodatkowo zostało udostępnione wiele kombinacji usług takich jak: aplikacja jako usługa (*ang. Software-as-a-Service*) (**SaaS**), Interfejs jako usługa (*ang. Infrastructure-as-a-Service*) (**IaaS**), Platforma jako usługa (*ang. Platform-as-a-Service*) (**PaaS**), dzięki czemu uzyskano platformę przyjazną użytkownikowi umożliwiając korzystanie z ponad 200 dostępnych usług. Dodatkowo płatność za platformę jest rozliczana tylko za zadaną przestrzeń oraz wykorzystaną moc obliczeniową.[28, 29, 30].



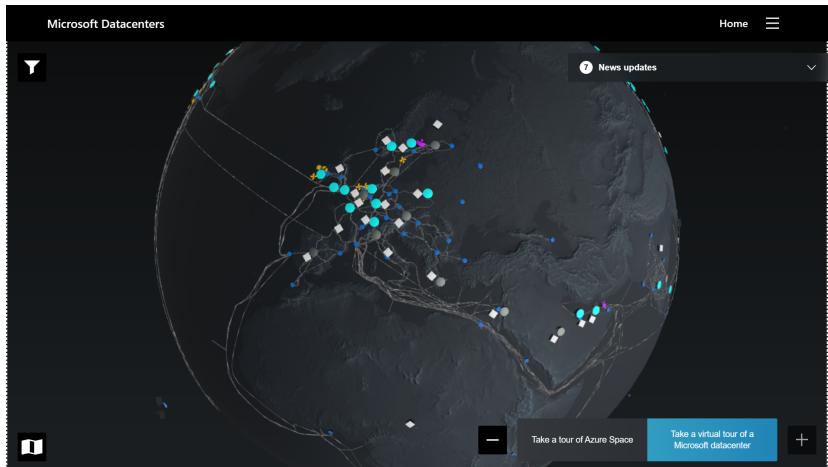
Rys. 5.1. Schemat podziału usług MS Azure
Źródło: [31]

Schemat 5.1 pokazuje jak obecnie podzielone są usługi oraz co jest udostępnione komercyjnie w ramach platformy Azure. Według schematu platforma podzielona jest na trzynaście obszarów, do których zaliczono między innymi bezpieczeństwo, zarządzanie danymi, usługi deweloperskie, analiza danych, platformy aplikacji.

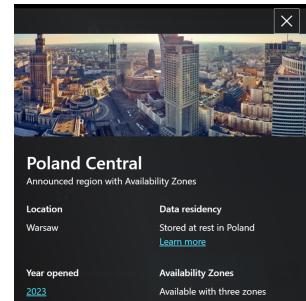
5.1. Infrastruktura

Infrastruktura globalna Azure składa się z dwóch części: fizycznej infrastruktury oraz globalnej łączności. Infrastruktura fizyczna składa się z ponad 200 centrów danych na całym świecie, połączonych w jedną globalną sieć. Dzięki czemu Azure umożliwia wysoką skalowalność i dostępność swoich rozwiązań. Jednakże cały ruch sieciowy jest utrzymywany wewnętrznie sieci Microsoft dzięki czemu informacje o adresach IP i ruchu sieciowym nie trafiają do publicznej części internetu[32].

Na swojej stronie internetowej Microsoft udostępnia wirtualną mapę umożliwiającą zobaczenie na własne oczy jak rozległa jest sieć Microsoftu. Dzięki interaktywności jest możliwość uzyskania informacji o kraju oraz centrum danych, a także komplikacjach jakie wynikają z przepisów wewnętrznych, pokazuje to zdjcie ??.



(a) Globalna mapa infrastruktury sieciowej
Zródło: [33]



(b) Informacje o centrum danych
Zródło: [34]

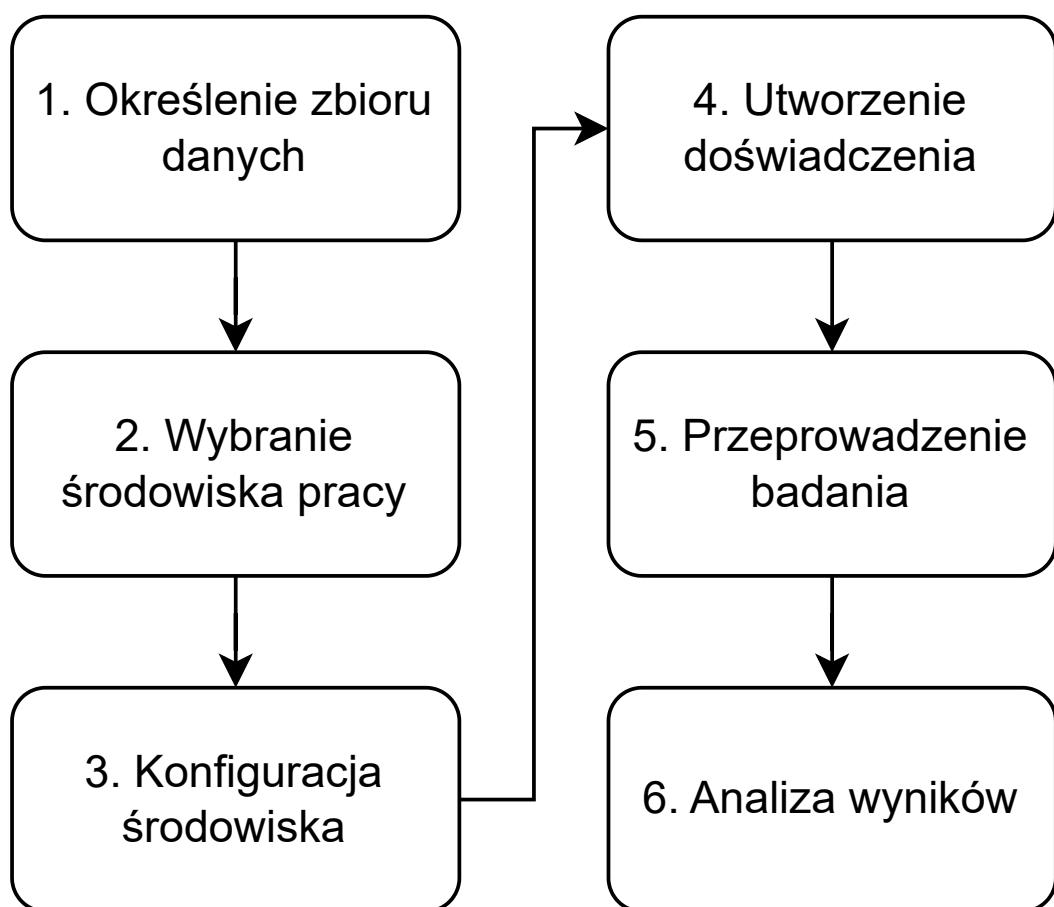
5.2. Machine Learning Studio

Azure Machine Learning Studio umożliwia łatwe i szybkie tworzenie wysoce wydajnych modeli uczenia maszynowego, a także zarządzanie nimi. Rozwiążanie wspiera pełen cykl życia kompleksowego uczenia maszynowego. Platforma umożliwia tworzenie potoków zadań, które połączone w jeden potok, wykonują poszczególne zadania w odpowiedniej kolejności. Dzięki modułowości modeli uzyskano rozwiązywanie wielokrotnego użytku, a w ramach jednego doświadczenia dany moduł, jeśli nie zostanie zmodyfikowany on, bądź zadania nad nim, zostaje ponownie użyty wynik danego modułu z poprzedniego doświadczenia. Dodatkowo poza predefiniowanymi operacjami można wykorzystać moduły języka Python/R. Dodatkowo można tworzyć rozwiązania w oparciu o "**Jupyter Notebook**", bądź wizualne narzędzie wykorzystujące wizualne układanie "*kafelk*" do tworzenia potoków zadań. Dodatkowo każde zadanie wykorzystuje wcześniej przygotowaną jednostkę obliczeniową, dzięki czemu można przewidzieć albo dostosować koszt korzystania z modelu. Umożliwione zostało również wdrażanie modeli jako punktów końcowych, co umożliwia komunikowanie się z nimi za pomocą REST API.

Microsoft umożliwia płatność jedynie za użytkowanie usług, co oznacza, że jeśli klaszter komputerowy był wykorzystywany jedynie przez 1 godzinę, to za tą jedną godzinę zostanie obciążony klient[35].

6. Opis doświadczenia

Przeprowadzone doświadczenie polega na porównaniu dostępnych w środowisku Microsoft Azure algorytmów klasyfikacji danych dwu-klasowych wraz z algorytmem stworzonym na potrzeby pracy inżynierskiej o tytule *"Wykorzystanie algorytmów genetycznych w systemach wykrywania intruzów w sieciach komputerowych"*[36] oraz z algorytmem DANet[37]. Doświadczenie przebiegało według **schematu 6.1**.



Rys. 6.1. Schemat przebiegu doświadczenia
 Źródło: Opracowanie własne

6.1. Dane

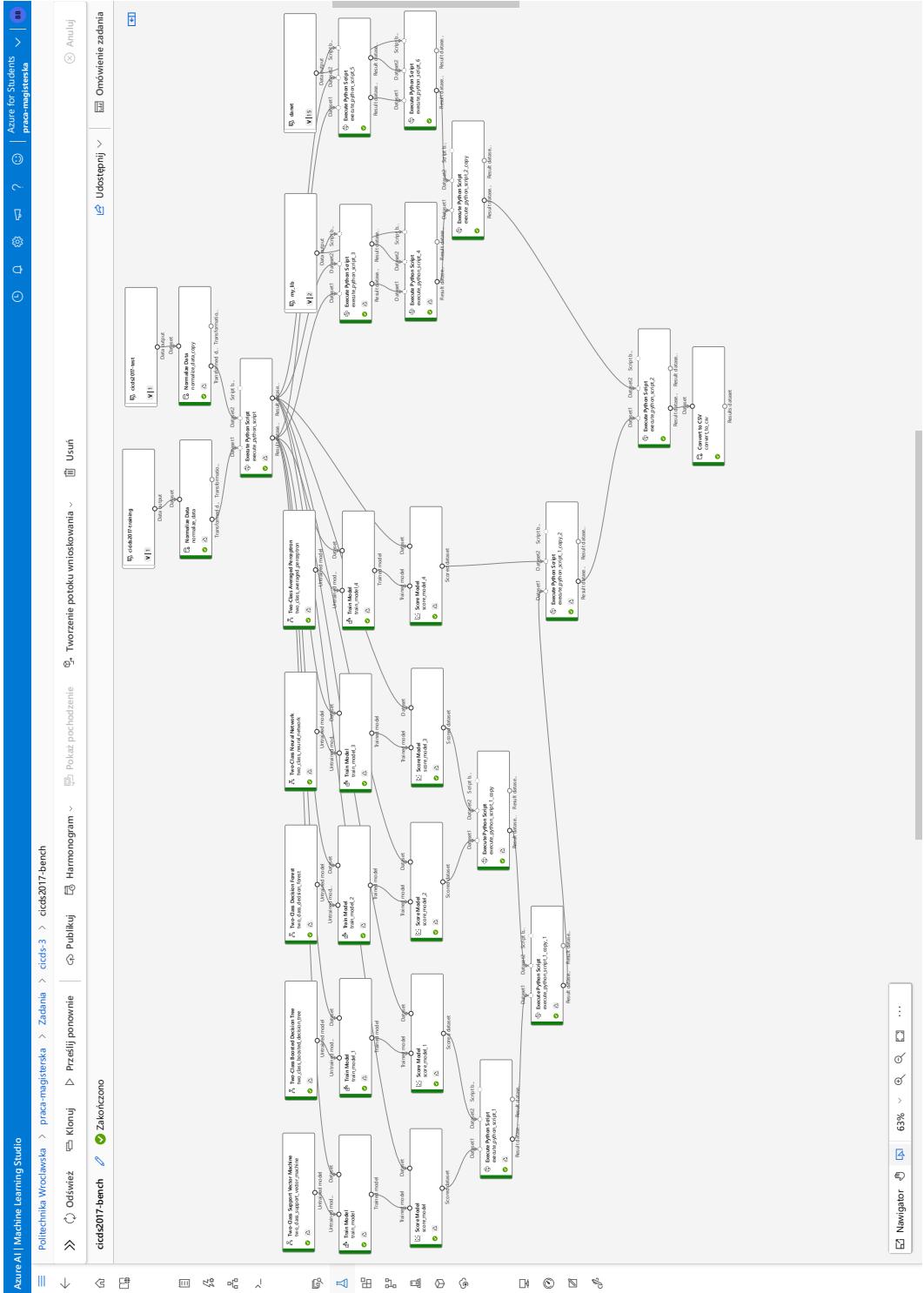
Zbiór danych został przygotowany przez Kanadyjski Instytut Cyberbezpieczeństwa działający przy Uniwersytecie Nowy Brunszwik za pomocą narzędzia CICFlowMeter[38].

Zbiór zawiera 79 cech ruchu sieciowego do których zaliczyć można: etykietę, czas trwania przesyłu, minimalną długość pakietu zwrotnego, maksymalną długość pakietu zwrotnego, port docelowy, długość pakietów. Zbiór pozwala na określenie czy ruch sieciowy jest życzliwy (ang. *BENING*), czy nieżyczliwy (różne możliwe formy ataku na sieć). Dodatkowo zbiór został podzielony na pięć dni roboczych: poniedziałek 3.07.2017 - piątek 7.07.2017. Dane z poniedziałku zawierają jedynie ruch życzliwy, zaś w pozostałe dni zostały zasymulowane ataki na sieć komputerową[36, 39].

6.2. Środowisko programistyczne

Jako środowisko programistyczne zostało wybrane Azure Machine Learning Studio z powodu możliwości uniezależnienia obliczeń od komputera lokalnego, dodatkowo platforma umożliwia łatwy sposób na tworzenie skomplikowanych potoków zadań, które składają się z komponentów wielokrotnego użytku. Każdy komponent uruchamia się w środowisku odizolowanym od pozostałych operacji. Dzieje się tak dzięki wykorzystaniu wielo węzłowych klastrów obliczeniowych, bazujących na oprogramowaniu Docker, klastry te mogą skalować się w zależności od potrzeb oraz dostępnej jednostki[40].

Całe doświadczenie zostało odwzorowane w graficznym potoku narzędzia "*Projektant*" oraz przedstawione na **zdjęciu 6.2.**



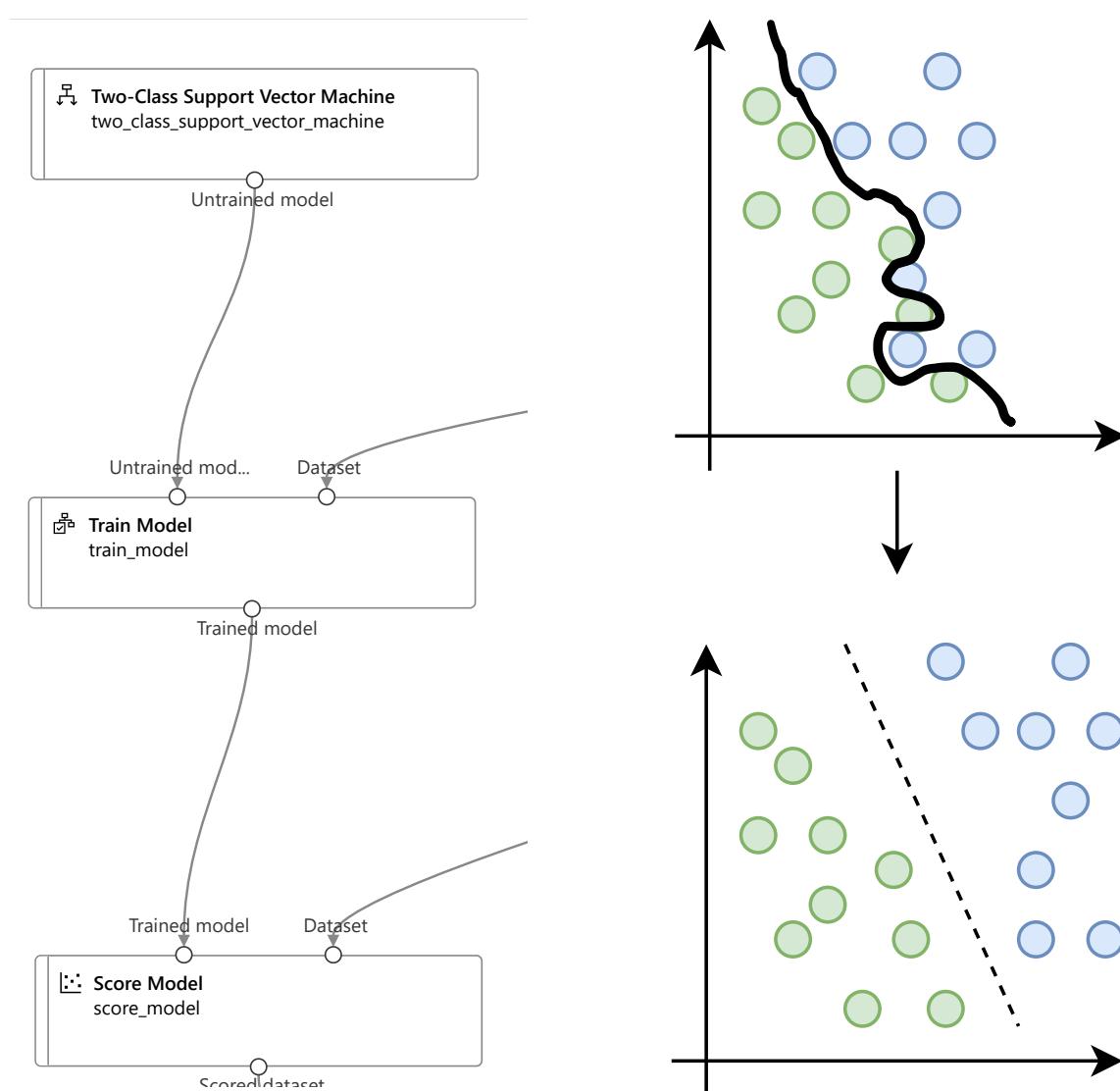
Rys. 6.2. Potok zadań
Źródło: Opracowanie własne

6.3. Algorytmy

W trakcie eksperymenty zastosowano różne algorytmy klasyfikacji danych. Charakterystyczną cechą tych algorytmów jest klasyfikacja ukierunkowana na 2 kategorie wejściowe. W tym wypadku są to kategorie ruchu sieciowego: [**BENIGN** (*pl. życzliwy*), **Other** (*pl. inne*)], gdzie inne to pozostałe typy ruchu sieciowego.

6.3.1. Two-Class Support Vector Machine

Algorytm SVM ma za zadanie znaleźć hiperpłaszczyznę w przestrzeni K-wymiarowej (K - liczba cech), która rozdziela zbiory punktów odpowiadających różnym klasom. W pierwszej kolejności szuka się separatora między klasami, a następnie przekształca się dane w taki sposób, by można przekształcić separator w hiperpłaszczyznę[41]. Sposób działania został zobrazowany za pomocą **wykresów 6.3b**. Część potoku 6.2 odpowiedzialnego za SVM to **schemat 6.3a**.



(a) Potok zadań dla modelu *Two-Class Support Vector Machine*

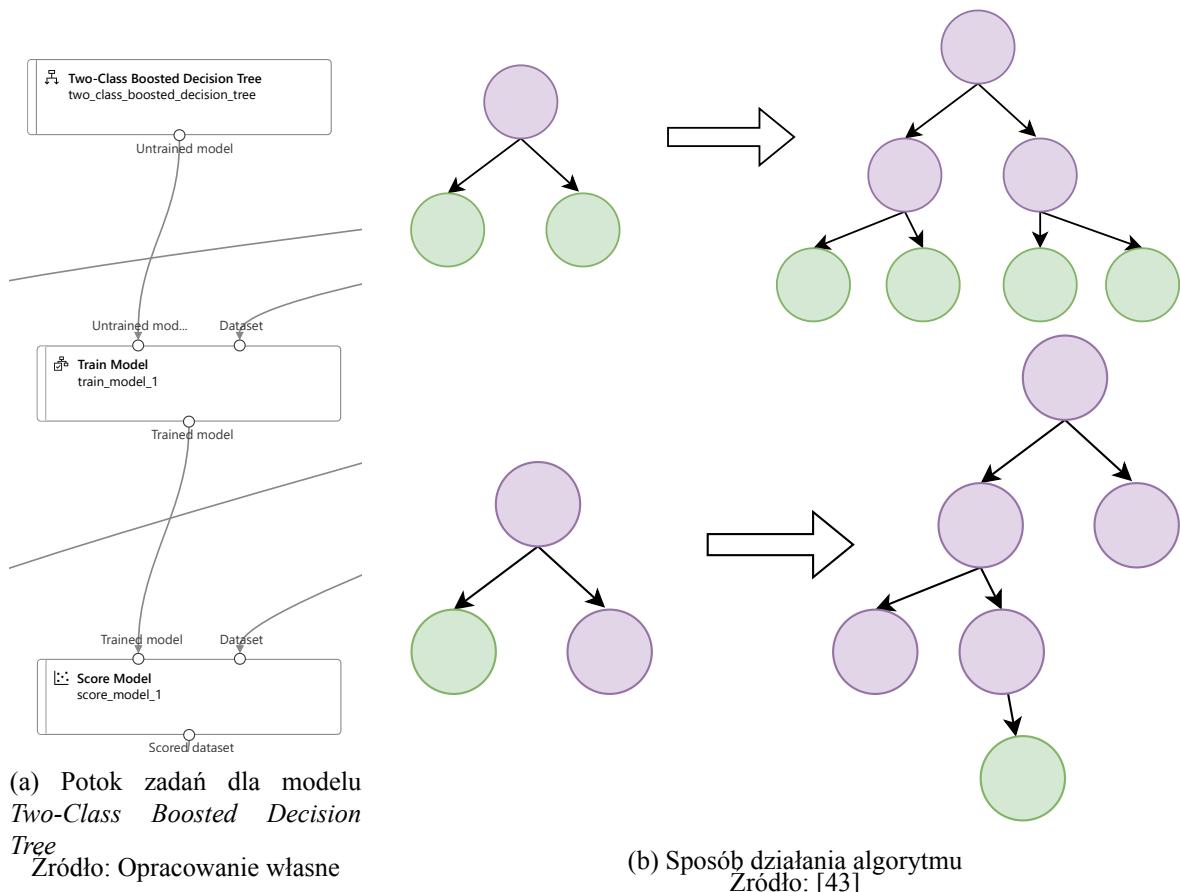
Zródło: Opracowanie własne

(b) Schemat SVM

Zródło: [42]

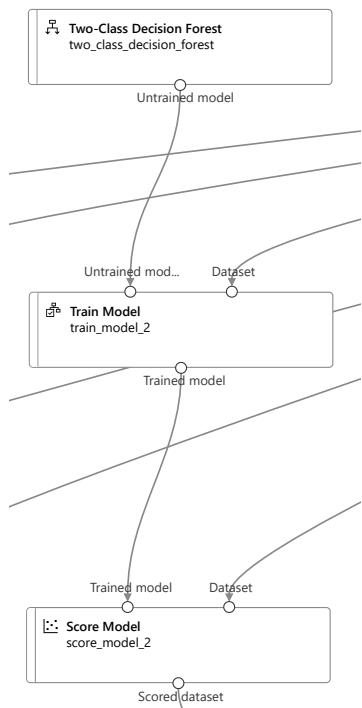
6.3.2. Two-Class Boosted Decision Tree

Jest to algorytm drzewa decyzyjnego oparty o algorytm LightGBM. Dzięki zastosowaniu tego podejścia algorytm oparty o drzewo decyzyjne działa szybciej oraz ma mniejszą złożoność obliczeniową. Algorytm ten działa na zasadzie doboru odpowiedniego liścia zamiast jak w przypadku klasycznych algorytmów opartych na drzewie, wyboru odpowiedniej warstwy warstwy[43]. Sposób podejścia liściastego został ukazany na **schemacie 6.4b**. Model wykorzystywany w Azure ML został ukazany na **rysunku 6.4a**.



6.3.3. Two-Class Decision Forest

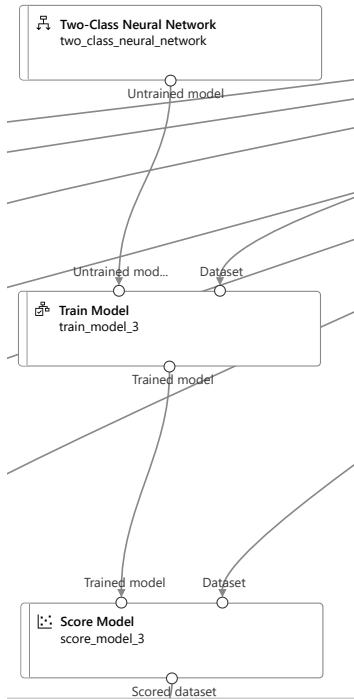
Las decyzyjny to algorytm którego wynik opiera się o agregację wyników wielu drzew decyzyjnych. Uzyskanie wyniku zależy od algorytmu trenowania lasu. Przykładowo w klasyfikacji losowym lasem wieloklasowym (*ang. Multi-class random forest classification*), każde drzewo głosuje na jedną klasę i ta, która zostanie wybrana większością głosów zostaje uznana za wynikową[44]. Model wykorzystany w Azure ML jest pokazany na **modelu 6.5**



Rys. 6.5. Potok zadań dla modelu *Two-Class Decision Forest*
 Źródło: Opracowanie własne

6.3.4. Two-class Neural Network

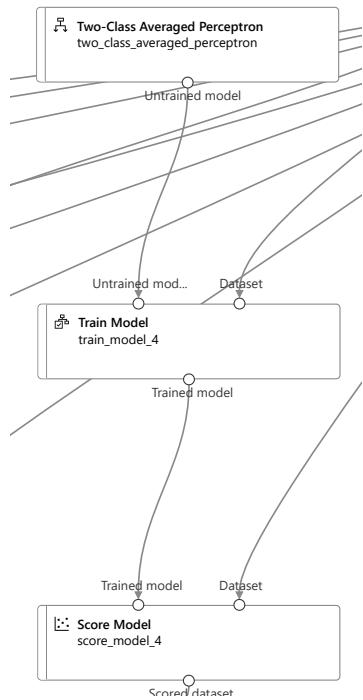
Jest to sieć neuronowa składająca się w tym wypadku z warstwy wejściowej, trzech warstw ukrytych każda po 100 węzłów, oraz z warstwy wyjściowej. Przykładowa sieć neuronowa została zobrazowana na **schemacie 2.7**. Moduł wykorzystany w Azure ML ukazano na **rysunku 6.6**.



Rys. 6.6. Potok zadań dla modelu *Two-Class Neural Network*
 Źródło: Opracowanie własne

6.3.5. Two-Class Average Perceptron

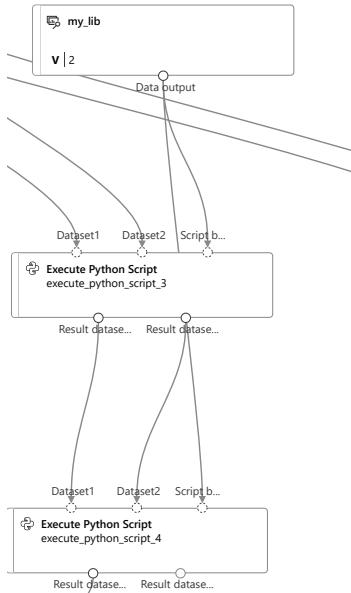
Jest to najprostsza odmiana sieci neuronowej czyli pojedynczy perceptron, który jest matematycznym modelem neuronu. Składa się on z n wejść, takiej samej ilości wag, progu Θ , sumatora, funkcji aktywującej i wyjścia. Został zobrazowany na **schemacie 2.6**. Może służyć za prosty klasyfikator binarny, albo za regresor. Model wykorzystany w Azure ML ukazano na **zdjęciu 6.7**.



Rys. 6.7. Potok zadań dla modelu *Two-Class Average Perceptron*
 Źródło: Opracowanie własne

6.3.6. Autorskie rozwiązanie

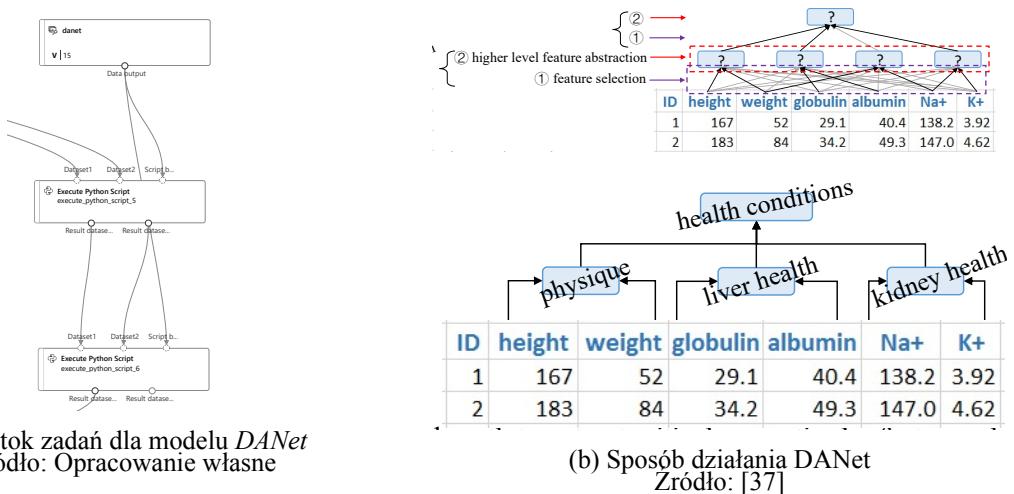
Algorytm ten polega na połączeniu algorytmu genetycznego (GA) wraz z klasyfikatorem naiwnym Bayesa wykorzystującego rozkład Gaussa (GNB). Zadaniem algorytmu genetycznego jest znalezienie naistotniejszych cech w zbiorze tabelarycznym, które pozwolą na zmniejszenie wymiarowości danych, oraz na zmniejszenie kosztów obsługi samego klasyfikatora w późniejszych etapach testowania, ze względu na zmniejszoną ilość danych wymaganych do przetworzenia. GA wykorzystywał w metodzie **fitness** algorytm GNB w celu określenia dopasowania danych. Zadaniem GNB było znalezienie najlepszej dostępnej kombinacji cech, które pozwalały na uzyskanie najlepszego dopasowania [36]. Model wykorzystywany w Azure ML różni się od gotowych modeli tym, że dołączono do niego bibliotekę napisaną w języku python, która zawiera kod wykorzystywany w pracy inżynierskiej autora[45].



Rys. 6.8. Potok zadań dla modelu
 Źródło: Opracowanie własne

6.3.7. DANet

Twórcy tego algorytmu wprowadzają dodatkową warstwę abstrakcyjną o nazwie "*Abstract Layer*". Wartswy te budują sieć o nazwie "*Deep Abstract Network*" (DANet). Zadanie dodatkowych warstw jest grupowaniem cech w skorelowanych zbiorach. Zbiory te budują się powiązań między sobą w formie sieci semantycznej. Gdy sieć semantyczna jest zbudowana w ostatnim kroku wykonywana jest klasyfikacja w trzywarstwowej sieci perceptronów (ang. *Multilayer Perceptron network*) (MLP)[37, 46]. Model znajdujący się w Azure ML został przedstawiony na **zdjęciu 6.9a**, zaś sposób działania ukazano na **schematach 6.9b**.



7. Przebieg badań

Doświadczenie polegało na analizie porównawczej sprawności algorytmów opisanych w **rozdziale 6 w sekcji 6.3**. Celem doświadczenia było określenie jakości algorytmu utworzonego w ramach pracy inżynierskiej autora[36]. Szczegółowa metodologia badawcza została określona w **podrozdziale 7.1**.

7.1. Metodologia badawcza

Przyjęta w projekcie metodologia badawcza została określona w poniższej **tabeli 7.1**. Przyjęta metodologia ma za zadanie określić jakość porównywanego algorytmu.

Tabela 7.1. Metodologia badawcza
Źródło: Opracowanie własne

Problem badawczy:

Czy algorytm klasyfikacji danych utworzony w ramach pracy inżynierskiej może konkurować z rozwiązaniami dostępnymi w środowiskach komercyjnych

Pytania badawcze:

1. Czy algorytm jest konkurencyjny pod względem wybranych metry:
 - dokładność algorytmu
 - czas działania
 - precyzja
 - czułość
 - f1
 - auc

Hipotezy:

1. Nie ma istotnej różnicy w wynikach próby testowej i treningowej.
2. Nie ma istotnej różnicy w uzyskanej "dokładności" między algorytmami.
3. Nie ma istotnej różnicy w uzyskanej "czułości" między algorytmami.
4. Nie ma istotnej różnicy w uzyskanej "precyzyji" między algorytmami.
5. Nie ma istotnej różnicy w uzyskanym "auc" między algorytmami.
6. Nie ma istotnej różnicy w uzyskanej "f1" między algorytmami.
7. Nie ma istotnej różnicy w uzyskanym "czasie działania" między algorytmami.

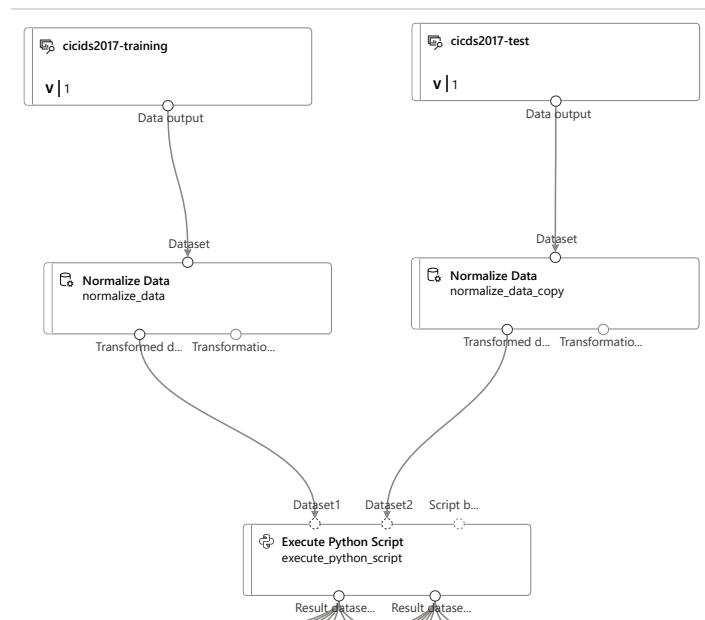
7.2. Przygotowanie platformy

Do badań wykorzystano narzędzie "Projektant" znajdujące się na platformie "Azure Machine Learning Studio" (Azure ML). Narzędzie to umożliwiło utworzenie interaktywnego potoku zadań. Potok ten składa się z kilku części:

- Przygotowanie i obróbka zbiorów danych
- Trenowanie oraz testowanie algorytmów klasyfikacji danych
- Utworzenie tabeli porównawczej dla wyników poszczególnych algorytmów (**obraz 6.2**).

7.2.1. Przygotowanie danych

W pierwszym kroku dane zostały znormalizowane za pomocą metody **MinMax**, która przekształca dane numeryczne do wartości w zakresie 0, 1. W kolejnym kroku za pomocą języka Python oraz bibliotek Pandas oraz Numpy zostają zamienione etykiety słowne na wartości **0** i **1** oraz następuje zamiana wartości $[NaN, -inf, inf]$ na cyfrę 0. Cały proces został zobrazowany na **diagramie 7.1**



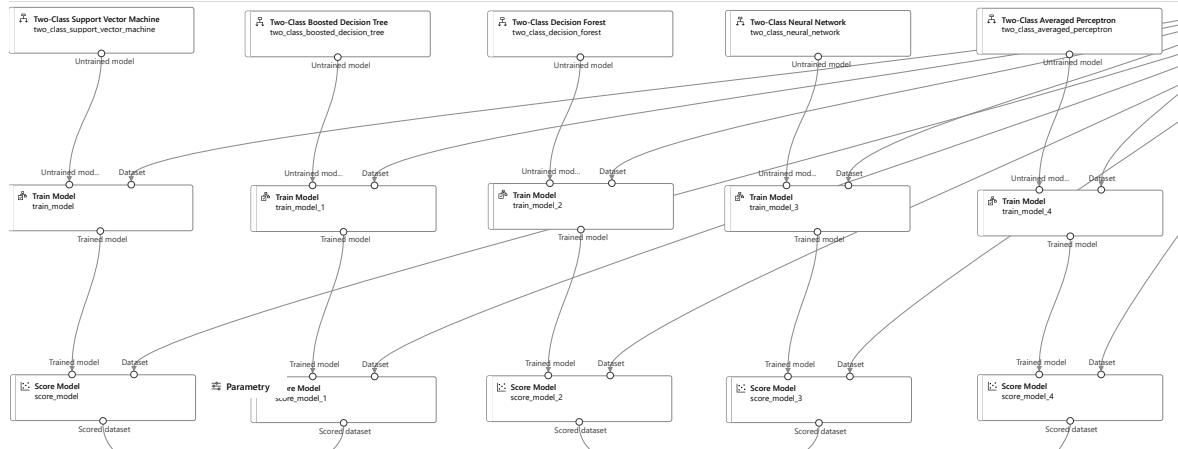
Rys. 7.1. Potok normalizacji danych
Źródło: Opracowanie własne

7.2.2. Trenowanie oraz testowanie algorytmów

Kolejną grupą zadań widoczną w potoku są te związane z trenowaniem i testowaniem poszczególnych algorytmów opisanych w **rozdziale 6**. Każdy test składa się 3 kafelek. W przypadku algorytmów dostarczonych wraz z platformą Azure ML są to:

- **model klasyfikujący** - odpowiada za przygotowanie algorytmu klasyfikacyjnego
- **blok treningowy** - tworzy wytrenowany model, za pomocą połączonego zbioru danych
- **blok ewaluacyjny** - sprawdza wcześniej wytrenowany model za pomocą powiązanego zbioru danych.

Potok zadań wykorzystujący algorytmy dostarczone przez Microsoft Azure został ukazany na **schemacie 7.2**

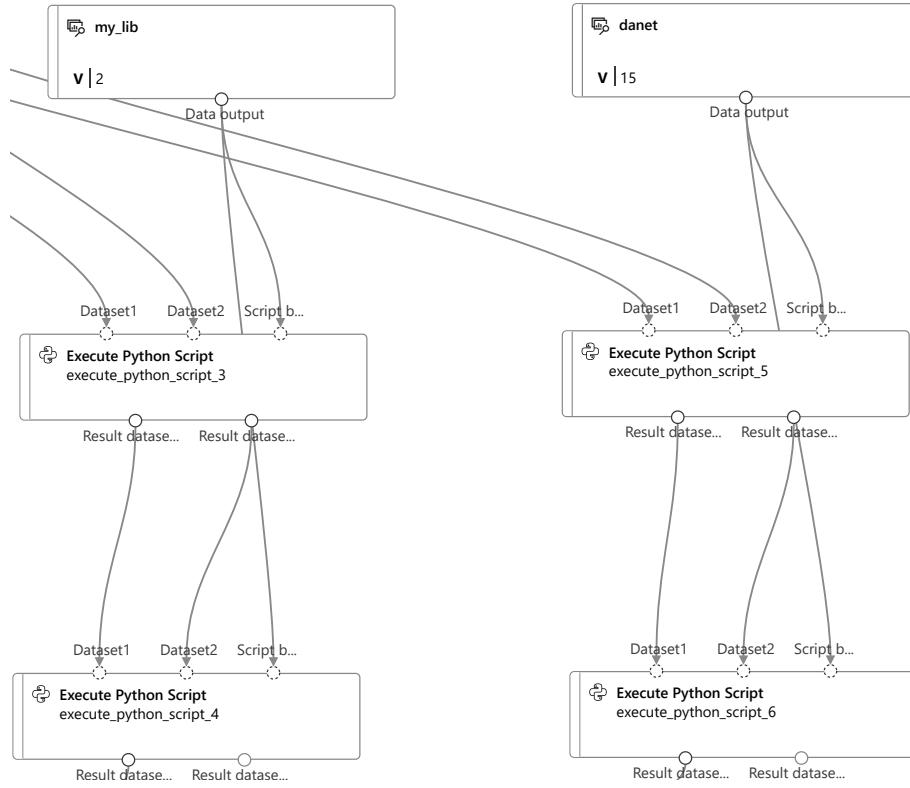


Rys. 7.2. Potok zadań dla algorytmów klasyfikacyjnych
Źródło: Opracowanie własne

Algorytmy dostarczone w ramach pracy badawczej składają się z:

- **biblioteka Python** - archiwum o rozszerzeniu .zip, które zawiera w sobie odpowiednie pliki napisane w języku Python
- **blok treningowy** - wykorzystuje dostarczoną bibliotekę do wytrenowania modelu oraz zapisania na platformie Azure najlepszego uzyskanego wyniku za pomocą powiązanego zbioru danych
- **blok ewaluacyjny** - wykorzystuje dostarczoną bibliotekę do ewaluacji algorytmu za pomocą połączonego zbioru danych

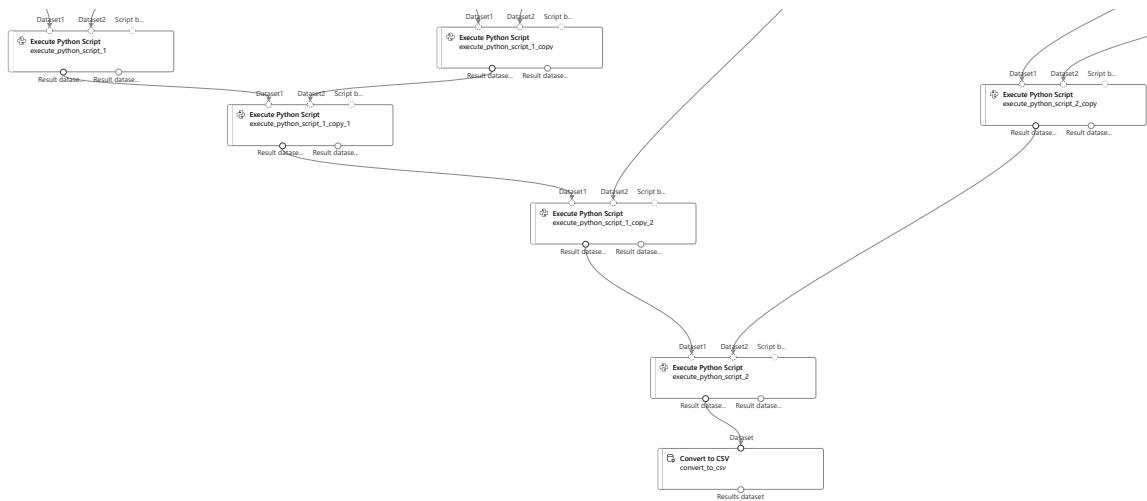
Potok zadań dla algorytmów niestandardowych został ukazany na **rysunku 7.3**



Rys. 7.3. Potok zadań dla algorytmów klasyfikacyjnych
Źródło: Opracowanie własne

7.2.3. Utworzenie tabeli porównawczej

Kolejną częścią zadań jest zebranie wyników poszczególnych algorytmów oraz połączenie ich w jedną całość. Wykorzystano do tego moduły języka Python, które zwracają przetwożone wyniki oraz łączą je w jedną tabelę zbiorczą, co pokazano na **rysunku 7.4.**



Rys. 7.4. Moduły odpowiedzialne za przetwożenie wyników
Źródło: Opracowanie własne

7.3. Weryfikacja potoku

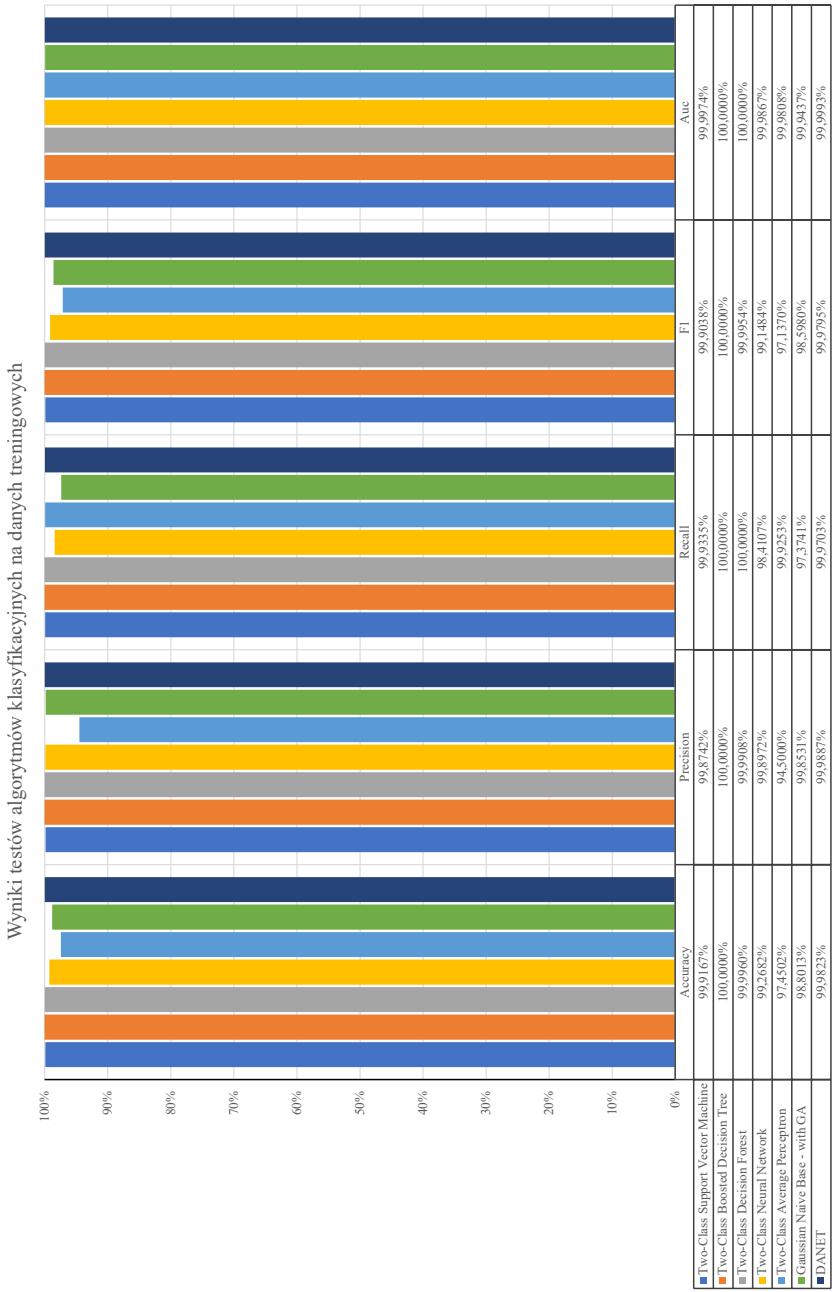
Aby zweryfikować działanie całego procesu wykorzystano znormalizowane dane treningowe do wytrenowania oraz przetestowania działania algorytmów klasyfikacyjnych. Cały proces trwał **"1 dzień 10 godzin 55 minut 53 sekundy"**. Wyniki tych działań widać na **rysunku 7.5**. Analizując wykres można zauważyć, że uzyskane wyniki znajdują się w przedziale [94%, 100%] w każdej metryce co pokazuje jakość każdego z algorytmów, a także to, że algorytmy poradziły sobie niemal bezbłędnie w rozpoznawaniu ruchu sieciowego, na którym były uczone. Zbiór, który wykorzystano do trenowania oraz testowania danych zawierał w sobie 225805 wpisów z czego 97718 należało do klasy **"1"**, zaś 128087 należało do klasy **"0"**.

Tabela 7.2. Liczba elementów przynależących do danej klasy w zbiorze treningowym

Źródło: Opracowanie własne

Klasa	Liczba wystąpień
1	97718
0	128027
Suma	225805

Bazując na tym zbiorze oraz uzyskanych wynikach udało się udowodnić poprawność działania procesu klasyfikacji wieloma algorytmami genetycznymi.



Rys. 7.5. Wyniki testów algorytmów klasyfikacyjnych na danych treningowych
 Źródło: Opracowanie własne

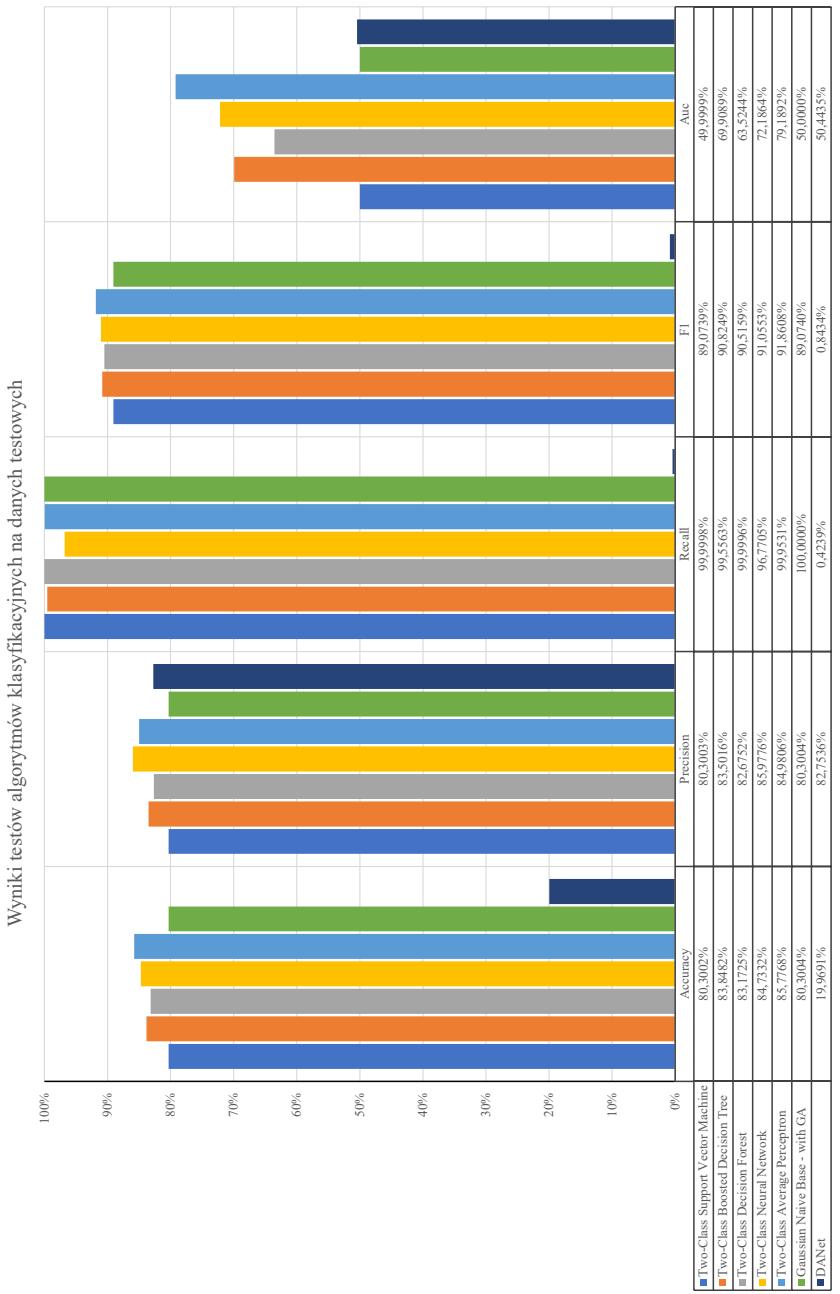
7.4. Próba badawcza

Aby uzyskać realne wyniki podczas porównywania poszczególnych algorytmów zastosowano zbiór treningowy opisany w **tabeli 7.2** oraz zbiór testowy, który zawierał 2273097 wpisów należących do klasy "1" oraz 557646 wpisów należących do klasy "0" co sumarycznie daje: 2830743 wpisów tak jak to zostało pokazane w **tabeli 7.3**. Pomiarы testowe powtórzono 2 razy dzięki czemu uzyskano 3 próby badawcze.

Tabela 7.3. Liczba elementów przynależących do danej klasy w zbiorze testowym

Źródło: Opracowanie własne

Klasa	Liczba wystąpień
1	2273097
0	557646
Suma	2830743



Rys. 7.6. Wyniki testów algorytmów klasyfikacyjnych na danych testowych
 Źródło: Opracowanie własne

Poniżej zostały przedstawione wyniki zbiorcze dla poszczególnych metryk. Dodatkowo przedstawiono również wynik pomiaru treningowego, który w większości przypadków jest wyższy od danych testowych. Co prawdopodobnie jest spowodowane różnicą w ilości danych testowych i treningowych. Dodatkowo w każdej kolumnie oznaczono kolorem zielonym najwyższy wynik dla danej metryki, a kolorem czerwonym najniższy wynik dla danej metryki.

7.4.1. Dopasowanie

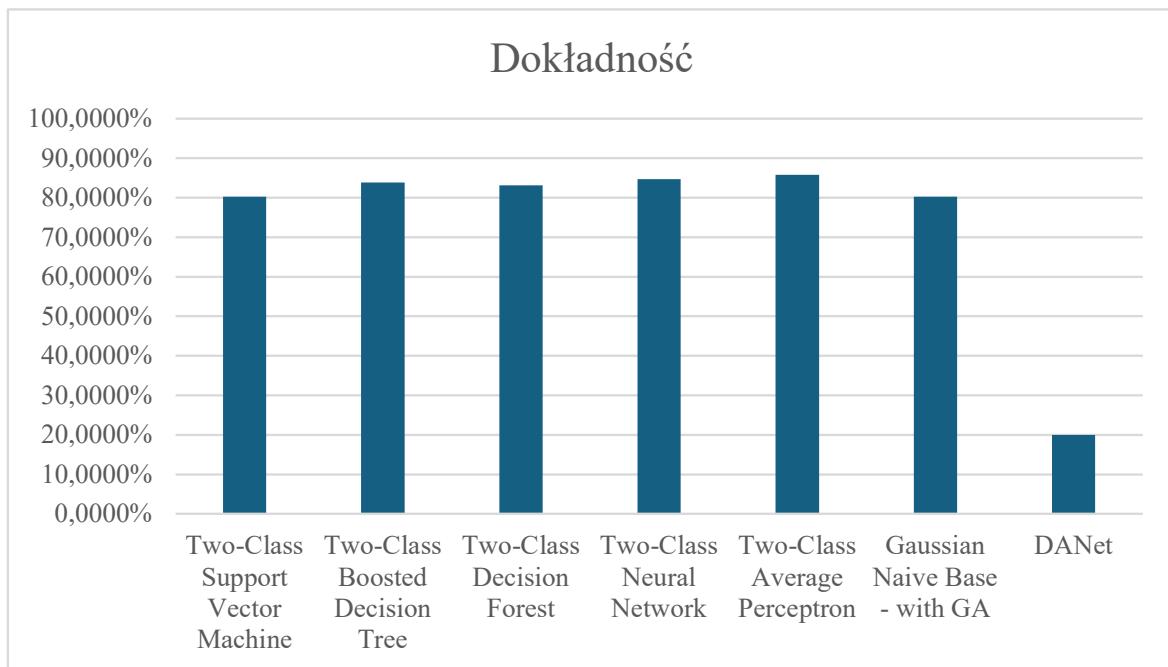
Najlepszy wynik dopasowania dla danych treningowych uzyskał algorytm *Two-Class Average Perceptron*, który poprawnie rozpoznał 85, 7768% próbek. Najgorszy wynik uzyskał algorytm *DANet* z dopasowaniem rzędu: 19, 9691%. Dla próby treningowej najlepszy wynik uzyskał *Two-Class Boosted Decision Tree* z wynikiem 100, 00%, a najgorszy *Two-Class Average Perceptron* z wynikiem 97, 4502%. Wyniki dopasowania dla poszczególnych prób zostały przedstawione na **tabeli 7.4** oraz na wykresie **wykresie 7.7**

Tabela 7.4. Wynik dopasowania algorytmów.

Kolorem zielonym określono najlepszy wynik w kolumnie. Kolorem czerwonym określono najgorszy wynik w kolumnie.

Źródło: Opracowanie własne

Algorytm	Wynik dopasowania			
	Próba 1	Próba 2	Próba 3	Próba testowa
Two-Class Support Vector Machine	80, 3002%	80, 3002%	80, 3002%	99, 9167%
Two-Class Boosted Decision Tree	83, 8482%	83, 8482%	83, 8482%	100, 0000%
Two-Class Decision Forest	83, 1725%	83, 1725%	83, 1725%	99, 9960%
Two-Class Neural Network	84, 7332%	84, 7332%	84, 7332%	99, 2682%
Two-Class Average Perceptron	85, 7768%	85, 7768%	85, 7768%	97, 4502%
Gaussian Naive Base - with GA	80, 3004%	80, 3004%	80, 3004%	98, 8013%
DANet	19, 9691%	19, 7899%	19, 7899%	99, 9823%



Rys. 7.7. Dokładność algorytmów
Źródło: Opracowanie własne

7.4.2. Precyzyja

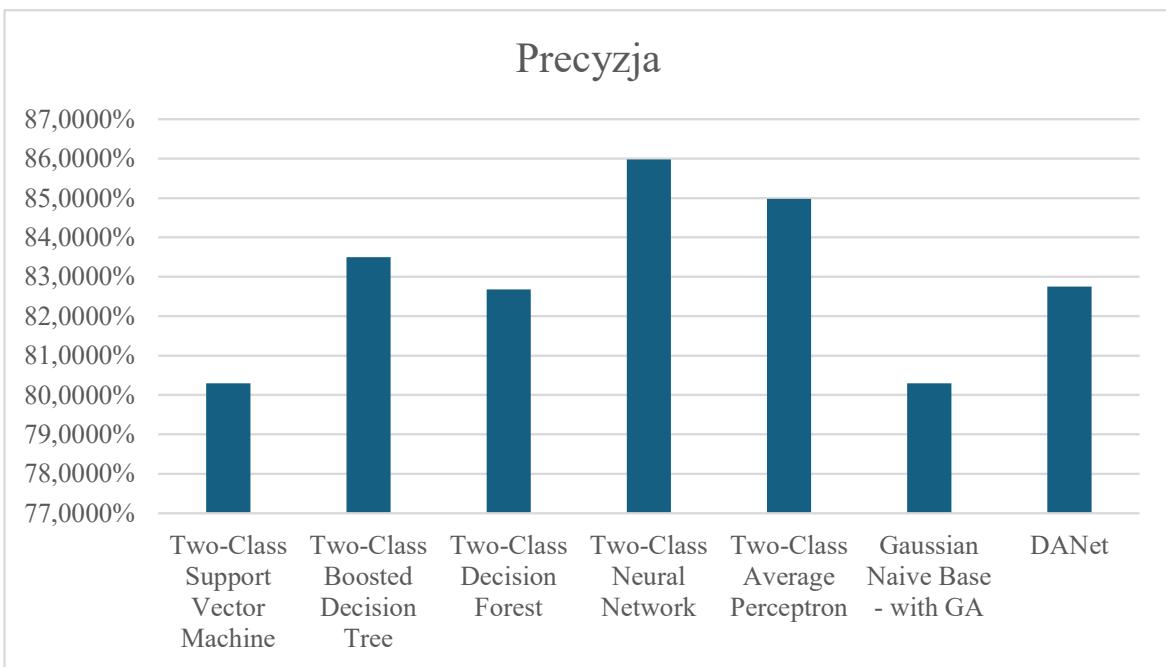
Najlepszy wynik precyzyji uzyskał algorytm "Two-Class Neural Network" (85, 9776%) co oznacza, że najlepiej poradził sobie z dopasowaniem ruchu klasy "1". Najsłabszy wynik uzyskał "Two-Class Support Vector Machine" (80, 3003%). GAGNB uzyskał 80, 3004% precyzyji. Wyniki są opisane w **tabeli 7.5** oraz na **wykresie 7.8**.

Tabela 7.5. Wynik precyzyji algorytmów.

Kolorem zielonym określono najlepszy wynik w kolumnie. Kolorem czerwonym określono najgorszy wynik w kolumnie.

Źródło: Opracowanie własne

Algorytm	Wynik precyzyji			
	Próba 1	Próba 2	Próba 3	Próba testowa
Two-Class Support Vector Machine	80,3003%	80,3003%	80,3003%	99,8742%
Two-Class Boosted Decision Tree	83,5016%	83,5016%	83,5016%	100,0000%
Two-Class Decision Forest	82,6752%	82,6752%	82,6752%	99,9908%
Two-Class Neural Network	85,9776%	85,9776%	85,9776%	99,8972%
Two-Class Average Perceptron	84,9806%	84,9806%	84,9806%	94,5000%
Gaussian Naive Base - with GA	85,7768%	80,3004%	80,3004%	99,8531%
DANet	82,7536%	85,9516%	85,9516%	99,9887%



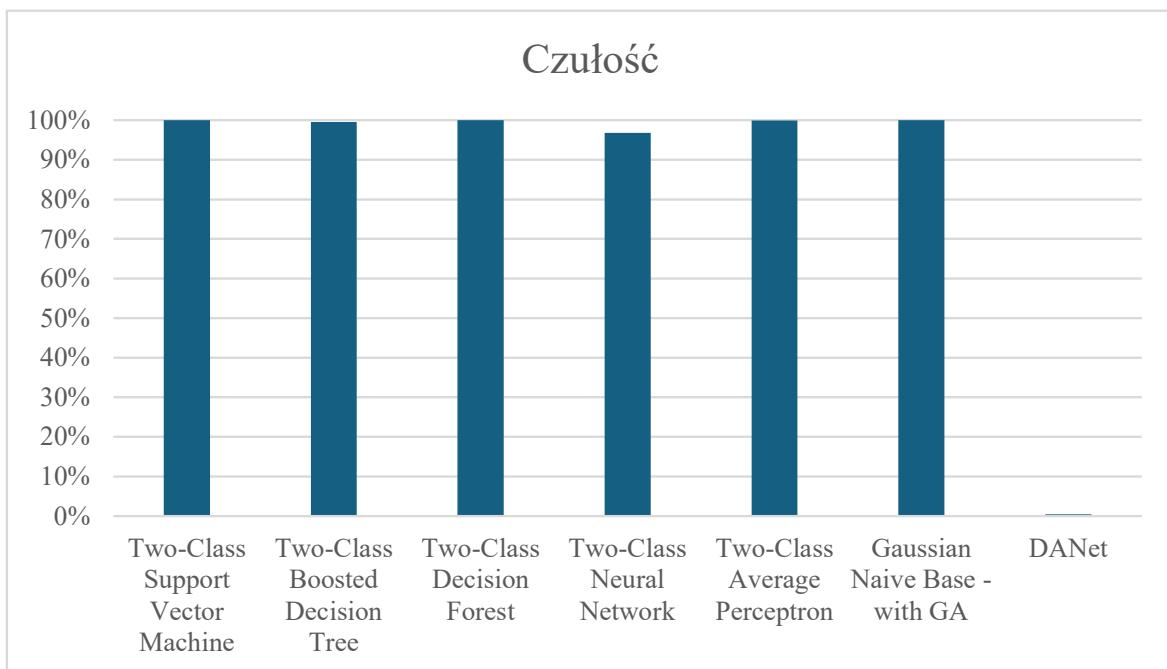
Rys. 7.8. Precyza algorytmów
Źródło: Opracowanie własne

7.4.3. Czułość

Wszystkie wpisy klasy "1" zostały poprawnie sklasyfikowane przez algorytm "Gaussian Naive Base - with GA" (100%), najsłabiej poradził sobie "Two-Class Neural Network" (96, 7705%). Wartości zostały przedstawione w **tabeli ??** oraz na **wykresie 7.9**.

Tabela 7.6. Wynik czułości algorytmów
Źródło: Opracowanie własne

Algorytm	Wynik czułości			
	Próba 1	Próba 2	Próba 3	Próba testowa
Two-Class Support Vector Machine	99, 9998%	99, 9998%	99, 9998%	99, 9335%
Two-Class Boosted Decision Tree	99, 5563%	99, 5563%	99, 5563%	100, 0000%
Two-Class Decision Forest	99, 9996%	99, 9996%	99, 9996%	100, 0000%
Two-Class Neural Network	96, 7705%	96, 7705%	96, 7705%	98, 4107%
Two-Class Average Perceptron	99, 9531%	99, 9531%	99, 9531%	99, 9253%
Gaussian Naive Base - with GA	100, 0000%	100, 0000%	100, 0000%	97, 3741%
DANet	0, 4239%	0, 1343%	0, 1343%	99, 9703%



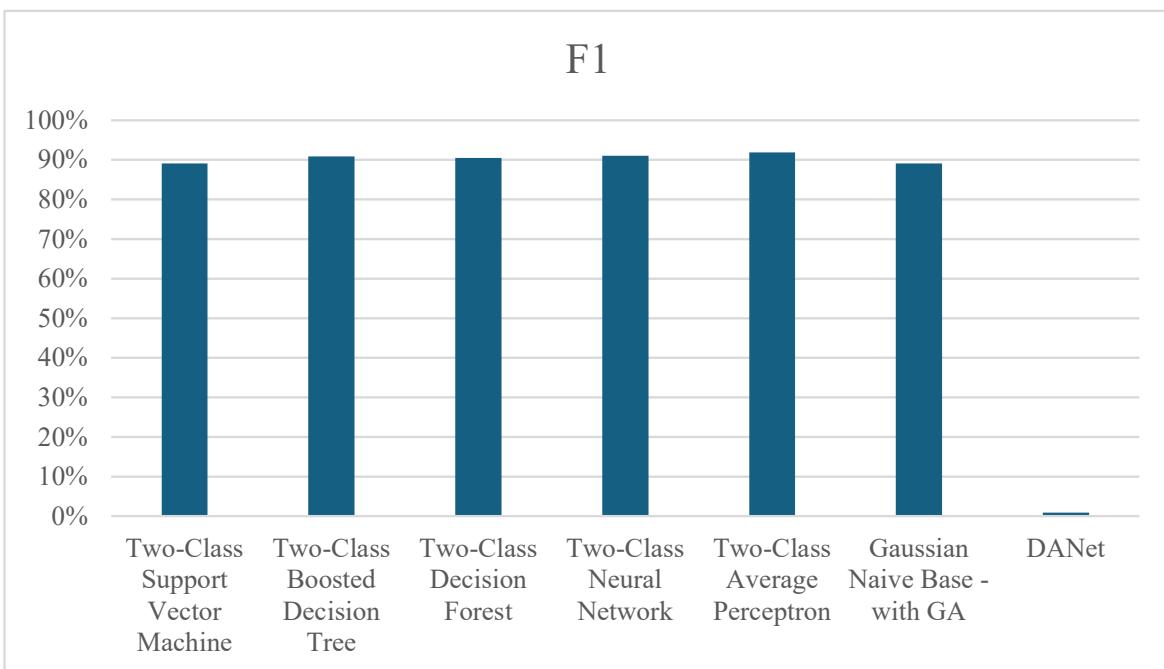
Rys. 7.9. Czułość algorytmów
Źródło: Opracowanie własne

7.4.4. F1

Największą wartość F1 uzyskał "Two-Class Average Perceptron" (91, 8608%). Najmniejszą uzyskał "Two-Class Support Vector Machine" (89, 0739%). GAGNB w tym teście uzyskał 89, 0740%. Wyniki zostały przedstawione w **tabeli 7.7** oraz na **wykresie 7.10**.

Tabela 7.7. Wynik F1 algorytmów
Źródło: Opracowanie własne

Algorytm	Wynik F1			
	Próba 1	Próba 2	Próba 3	Próba testowa
Two-Class Support Vector Machine	89, 0739%	89, 0739%	89, 0739%	99, 9038%
Two-Class Boosted Decision Tree	90, 8249%	90, 8249%	90, 8249%	100, 0000%
Two-Class Decision Forest	90, 5159%	90, 5159%	90, 5159%	99, 9954%
Two-Class Neural Network	91, 0553%	91, 0553%	91, 0553%	99, 1484%
Two-Class Average Perceptron	91, 8608%	91, 8608%	91, 8608%	97, 1370%
Gaussian Naive Base - with GA	89, 0740%	89, 0740%	89, 0740%	98, 5980%
DANet	0, 8434%	0, 2682%	0, 2682%	99, 9795%



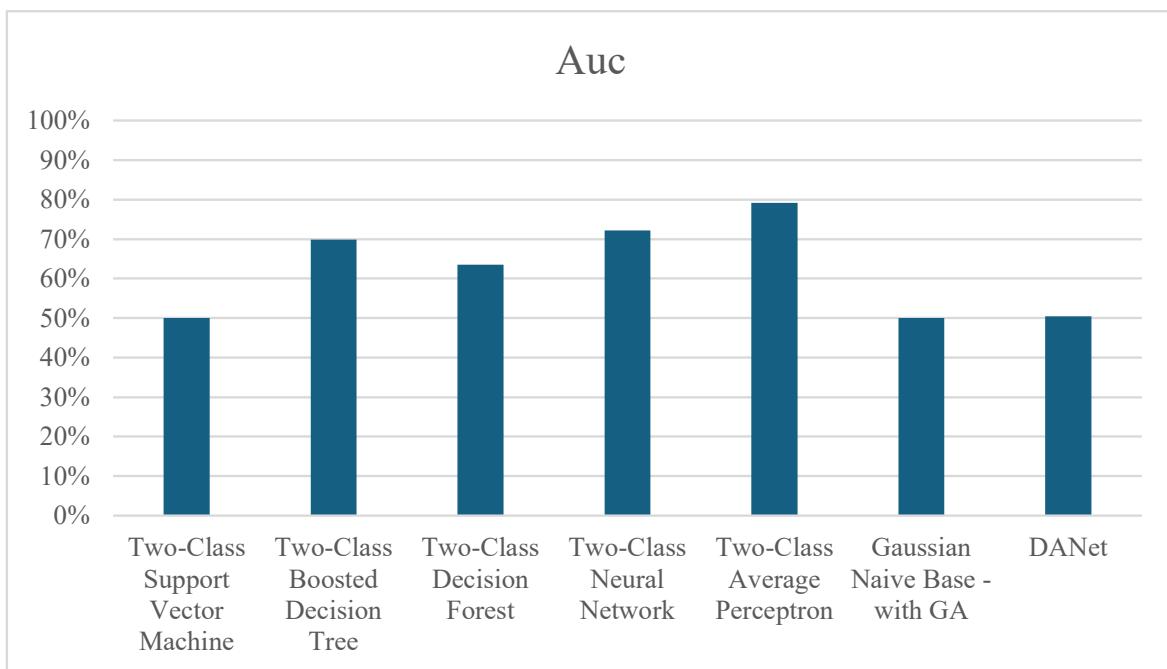
Rys. 7.10. F1 algorytmów
Źródło: Opracowanie własne

7.4.5. AUC

Analizując wynik AUC, można zauważyc, że najlepsza sprawność również uzyskał algorytm "Two-Class Average Perceptron" (79, 1292%). Najgorszy algorytm to znów algorytm wykorzystujący SVM (49, 9999%), wynik wskazał na to, że algorytm ten miał niższą sprawność niż próba losowa. W tym teście sprawność algorytmu GAGNB wyniosła 50%. Wyniki przedstawiono w tabeli 7.8 oraz na wykresie 7.11.

Tabela 7.8. Wynik AUC algorytmów
Źródło: Opracowanie własne

Algorytm	Wynik AUC			
	Próba 1	Próba 2	Próba 3	Próba testowa
Two-Class Support Vector Machine	49, 9999%	49, 9999%	49, 9999%	99, 9974%
Two-Class Boosted Decision Tree	69, 9089%	69, 9089%	69, 9089%	100, 0000%
Two-Class Decision Forest	63, 5244%	63, 5244%	63, 5244%	100, 0000%
Two-Class Neural Network	72, 1864%	72, 1864%	72, 1864%	99, 9867%
Two-Class Average Perceptron	79, 1892%	79, 1892%	79, 1892%	99, 9808%
Gaussian Naive Base - with GA	50, 0000%	50, 0000%	50, 0000%	99, 9437%
DANet	50, 4435%	76, 7282%	76, 7282%	99, 9993%



Rys. 7.11. AUC algorytmów
Źródło: Opracowanie własne

7.5. Analiza wyników

Przedział ufności dla dokładności algorytmów wyniósł 17,7218 dla $\alpha = 0.05$. Biorąc pod uwagę ten fakt można stwierdzić, że dokładność algorytmu DANet jest poniżej dolnej granicy. Dolna granica przedziału ufności wynosi 56,2925%, zaś DANet uzyskał 19,9691% oraz poprawnie zaklasyfikował jedynie 565273 wpisów.

7.6. Wnioski

Microsoft Wyszedł na przeciw potrzebom użytkowników dlatego przygotował zestaw prekonfigurowanych algorytmów klasyfikacyjnych, dzięki czemu możliwości narzędzia Azure ML pozwalają na odpowiadanie na konkretne potrzeby przy relatywnie niewielkich kosztach. Jednakże to nie oznacza, że tworzenie autorskich rozwiązań mija się z celem. Jak ukazano na **wykresie 7.6** autorskie rozwiązania również mają rację bytu. Wykorzystanie połączenia algorytmu genetycznego i klasyfikatora naiwnego Bayesa z rozkładem normalnym pozwala na uzyskanie zbliżonych wyników co algorytmu utworzonego przez giganta technologicznego. Różnica około 5 punktów procentowych między najlepszym algorytmem a algorytmem GAGNB ukazuje niewielką różnicę w jakości algorytmu. Dodatkowo wciąż trudno korzystać z rozwiązań takich jak DANet, które są nieprzetestowane na różnorakich zbiorach lecz na tych wcześniej specjalnie spreparowanych.

Dodatkowo korzystanie z tego typu prostych rozwiązań autorskich pozwala na

prototypownie rozwiązań biznesowych opartych o klasyfikację danych. Samo wykorzystanie algorytmu genetycznego z algorytmem GNB umożliwia skupienie się na tworzenie ogólnego rozwiązania bez wcześniejszej znajomości zbioru oraz korzystając z programów lokalnie nie ponosząc kosztów wykorzystania platformy chmurowej. Kolejnym atutem utworzonego przez autora rozwiązania jest zmniejszenie kosztów lokalnego użytkowania poprzez zmniejszenie wymiarowości zbioru danych do klasyfikacji poprzez wykorzystanie jedynej wytypowanych kolumn.

Doświadczenie to ukazuje, że wciąż należy próbować tworzyć wydajniejsze i dokładniejsze rozwiązania, lecz nie zaprzecza faktu iż rozwiązania ogólnie dostępne są na bardzo wysokim poziomie. Uzyskano dokładność z zakresu [80%, 85%] przy czym plik testowy był 12 krotnie większy od pliku treningowego.

8. Perspektywy rozwoju

Stworzony projekt jest jedynie silnikiem klasyfikacyjnym, który pozwala na wytrenowanie i wyłonienie najlepszego algorytmu do klasyfikacji danych. Dzięki możliwościom platformy Azure ML stworzenie całego środowiska testowego jest relatywnie tanie i nie wymaga wielu wyspecjalizowanych umiejętności. Bazując na wynikach i najlepszym klasyfikatorze, można utworzyć odpowiednie Przepływy służące do klasyfikacji danych wejściowych, do których dostęp może odbywać się za pomocą utworzonych Punktów Dostępowych (*ang. Endpoints*), a dzięki którym możliwa jest komunikacja za pomocą protokołu HTTPS (*ang. Hyper Text Transfer Protocol Secure*) i komunikacji typu REST (*ang. Representative State Transfer*).

Utworzony w ten sposób punkt dostępu może zostać wykorzystany w klasyfikacji ruchu sieciowego w niewielkich aplikacjach z dostępem do internetu. Pozwoliłoby to na realizację analizy danych w chmurze, co mogłoby przyspieszyć cały proces oraz utworzyć pojedyncze źródło prawdy dla wielu instancji aplikacji. A wykorzystanie dodatkowo konteneryzacji, którą zapewnia platforma Azure, cały proces mógłby zostać zoptymalizowany pod kątem wydajnościowym i lokalizacyjnym. Pojedyncze źródło prawdy jest zaletą wykorzystania *"zewnętrznego"* klasyfikatora, ponieważ zapewnia jednakowe wyniki klasyfikacji w poszczególnych instancjach samej aplikacji instalowanej na wielu urządzeniach. Pozwala to również na lepsze dostrajanie całego procesu, a wykorzystanie kolejnych wersji przepływów umożliwia utrzymywanie kopii zapasowych poszczególnych rozwiązań, dzięki czemu po wykryciu nieprawidłowości w nowych klasyfikacjach, można cofnąć proces do poprzedniej wersji.

Wykaz rysunków

2.1	Graficzne przedstawienie podziałów sztucznej inteligencji	9
2.2	Podział uczenia maszynowego	10
2.3	Uczenie nadzorowane	11
2.4	Uczenie nienadzorowane	11
2.5	Uczenie przez wzmacnianie	12
2.6	Schemat neuronu	13
2.7	Schemat sieci neuronowej	14
2.8	Schemat prostej głębszej sieci neuronowej	15
4.1	PowerApps od Microsoft	19
4.2	Wordpress.com	19
5.1	Schemat podziału usług MS Azure	24
6.1	Schemat przebiegu doświadczenia	27
6.2	Potok zadań	29
6.5	Potok zadań dla modelu <i>Two-Class Decision Forest</i>	32
6.6	Potok zadań dla modelu <i>Two-Class Neural Network</i>	33
6.7	Potok zadań dla modelu <i>Two-Class Average Perceptron</i>	34
6.8	Potok zadań dla modelu	35
7.1	Potok normalizacji danych	38
7.2	Potok zadań dla algorytmów klasyfikacyjnych	39
7.3	Potok zadań dla algorytmów klasyfikacyjnych	40
7.4	Moduły odpowiedzialne za przetwarzanie wyników	40
7.5	Wyniki testów algorytmów klasyfikacyjnych na danych treningowych	42
7.6	Wyniki testów algorytmów klasyfikacyjnych na danych testowych	44
7.7	Dokładność algorytmów	46
7.8	Precyzaja algorytmów	47
7.9	Czułość algorytmów	48
7.10	F1 algorytmów	49
7.11	AUC algorytmów	50

Wykaz tabel

1	Tabela skrótów	4
1.1	Założenia techniczne pracy dyplomowej	8
3.1	Macierz pomyłek	17
7.1	Metodologia badawcza	37
7.2	Liczba elementów przynależących do danej klasy w zbiorze treningowym	41
7.3	Liczba elementów przynależących do danej klasy w zbiorze testowym	43
7.4	Wynik dopasowania algorytmów	45
7.5	Wynik precyzji algorytmów	46
7.6	Wynik czułości algorytmów	47
7.7	Wynik F1 algorytmów	48
7.8	Wynik AUC algorytmów	49

Bibliografia

- [1] Microsoft. „*Microsoft Machine Learning Studio (classic)*”. 2022. URL: <https://studio.azureml.net/> (term. wiz. 2023-09-01).
- [2] The Pandas development team. „*pandas-dev/pandas: Pandas*”. Lut. 2019. doi: 9.5281/zenodo.3509134. URL: <https://doi.org/9.5281/zenodo.3509134>.
- [3] Wes McKinney. „*Data Structures for Statistical Computing in Python*”. W: *Proceedings of the 9th Python in Science Conference*. 2010, s. 56–61. doi: 10.25080/majora-92bf1922-00a.
- [4] UNB. „*CICIDS2017 | Kaggle*”. URL: <https://www.kaggle.com/datasets/cicdataset/cicids2017> (term. wiz. 2023-09-01).
- [5] Oxford University Press. „*intelligence, n., sense I*”. W: *Oxford English Dictionary*. Lip. 2023. doi: 10.1093/OED/3757635879.
- [6] „*Artificial Intelligence in Science*”. OECD, czer. 2023. doi: 10.1787/a8d820bd-en.
- [7] Batta Mahesh. „*Machine Learning Algorithms-A Review*”. W: *International Journal of Science and Research* (2018). issn: 2319-7064. doi: 10.21275/ART20203995.
- [8] Satavisa Pati. „*The Difference Between Artificial Intelligence and Machine Learning*”. W: *Emerj MI* (2018), s. 3–8.
- [9] Chun Zhang i in. „*Semi-supervised behavioral learning and its application*”. W: *Optik* 127.1 (2016), s. 376–382. issn: 00304026. doi: 10.1016/j.ijleo.2015.10.089.
- [10] Robert Koch. „*History of Machine Learning – A Journey through the Timeline*”. 2022. URL: <https://www.clickworker.com/customer-blog/history-of-machine-learning/> (term. wiz. 2023-09-02).
- [11] Alexander L. Fradkov. „*Early history of machine learning*”. W: *IFAC-PapersOnLine*. T. 53. 2. Elsevier, sty. 2020, s. 1385–1390. doi: 10.1016/j.ifacol.2020.12.1888.
- [12] Austin Pollard. „*What are neural networks?*” 1990. doi: 10.1108/eb007822. URL: <https://www.ibm.com/topics/neural-networks>.
- [13] Karolina Bartos. „*SIEĆ SOM JAKO PRZYKŁAD SIECI SAMOORGANIZUJĄCEJ SIĘ*”. W: (2012). issn: 1507-3866.
- [14] Algolytics. „*Jak ocenić jakość i poprawność modeli klasyfikacyjnych ? Część 4-Krzywa ROC*”. URL: <https://algolytics.pl/tutorial-jak-ocenic-jakosc-i-poprawnosc-modeli-klasyfikacyjnych-czesc-4-krzywa-roc/> (term. wiz. 2023-09-04).
- [15] Wordpress. „*WordPress.com: Build a Site, Sell Your Stuff*”. 2023. URL: <https://wordpress.com/> (term. wiz. 2023-09-04).

- [16] JoomlaORG. „*Joomla! - Content Management System to build websites*”. 2021. URL: <https://www.joomla.org/%20https://www.joomla.org/about-joomla.html> (term. wiz. 2023-09-04).
- [17] Wix. „*Free website builder | Create a free website*”. 2016. URL: <https://www.wix.com/> (term. wiz. 2023-09-04).
- [18] Microsoft. „*PowerApps*”. 2023. URL: <https://guidedtour.microsoft.com/guidedtour/scenarios/power-apps/2.2.png> (term. wiz. 2023-09-04).
- [19] Wordpress. „*Playground Demo*”. 2023. URL: <https://developer.wordpress.org/playground/demo/> (term. wiz. 2023-09-04).
- [20] Alexander C. Bock i Ulrich Frank. „*Low-Code Platform*”. W: *Business and Information Systems Engineering* 63.6 (grud. 2021), s. 733–740. ISSN: 18670202. doi: [10.1007/S12599-021-00726-8/FIGURES/1](https://doi.org/10.1007/S12599-021-00726-8/FIGURES/1).
- [21] Martin Hirzel. „*Low-Code Programming Models*”. W: (maj 2022). arXiv: 2205.02282.
- [22] Microsoft. „*Co to jest usługa Power Apps? - Power Apps | Microsoft Learn*”. URL: <https://learn.microsoft.com/pl-pl/power-apps/powerapps-overview> (term. wiz. 2023-09-05).
- [23] Microsoft. „*Omówienie tworzenia aplikacji kanw - Power Apps | Microsoft Learn*”. URL: <https://learn.microsoft.com/pl-pl/power-apps/maker/canvas-apps/getting-started> (term. wiz. 2023-09-05).
- [24] Microsoft. „*Omówienie tworzenia aplikacji opartej na modelu z Power Apps - Power Apps | Microsoft Learn*”. URL: <https://learn.microsoft.com/pl-pl/power-apps/maker/model-driven-apps/model-driven-app-overview> (term. wiz. 2023-09-05).
- [25] Microsoft. „*Omówienie kart dla usługi Power Apps - Power Apps | Microsoft Learn*”. URL: <https://learn.microsoft.com/pl-pl/power-apps/cards/overview> (term. wiz. 2023-09-05).
- [26] AmazonQuickSight. „*Business Intelligence Service – Amazon QuickSight – AWS*”. URL: <https://aws.amazon.com/quicksight/> (term. wiz. 2023-09-05).
- [27] GoogleAppSheet. „*Google AppSheet | Build apps with no code*”. URL: <https://about.appspot.com/home/> (term. wiz. 2023-09-05).
- [28] Abandy Roosevelt. „*History of Microsoft Azure*”. 2022. URL: <https://techcommunity.microsoft.com/t5/educator-developer-blog/the-history-of-microsoft-azure/ba-p/3574204%20https://techcommunity.microsoft.com/t5/educator-developer-blog/the-history-of-microsoft-azure/ba-p/3574204#> (term. wiz. 2023-09-08).
- [29] Microsoft Azure. „*Poznaj platformę Azure*”. URL: <https://azure.microsoft.com/pl-pl/explore/> (term. wiz. 2023-09-06).
- [30] Datashift. „*Microsoft Azure*”. URL: <https://www.datashift.eu/technology/microsoft-azure/?fbclid=IwAR262r9Kdc0oeF1I8PmCmCuu-P6-5VSHnoKfPyjTJTsEmOkIgmVmfsuuIS8> (term. wiz. 2023-09-08).
- [31] Datashift. „*MS Azure.png*”. URL: <https://cdn.nimbu.io/s/znvdo1j/pages/8g7p2fo/MS%20Azure.png?33zmiw4> (term. wiz. 2023-09-08).
- [32] Microsoft Azure. „*Infrastruktura globalna*”. URL: <https://azure.microsoft.com/pl-pl/explore/global-infrastructure/> (term. wiz. 2023-09-06).

- [33] Microsoft Azure. „*Azure global infrastructure experience*”. URL: <https://datacenters.microsoft.com/globe/explore> (term. wiz. 2023-09-08).
- [34] Microsoft Azure. „*Azure global infrastructure experience*”. URL: https://datacenters.microsoft.com/globe/explore?info=region_polandcentral (term. wiz. 2023-09-08).
- [35] Microsoft Azure. „*Azure Machine Learning — uczenie maszynowe jako usługa*”. URL: <https://azure.microsoft.com/pl-pl/products/machine-learning> (term. wiz. 2023-09-08).
- [36] Bartosz Błyszcz. „*Wykorzystanie algorytmów genetycznych w systemach wykrywania intruzów w sieciach komputerowych*”. Praca Inżynierska. Kraków: Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, wrz. 2022.
- [37] Jintai Chen i in. „*DANETs: Deep Abstract Networks for Tabular Data Classification and Regression*”. W: *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022*. T. 36. Association for the Advancement of Artificial Intelligence, grud. 2022, s. 3930–3938. ISBN: 1577358767. doi: [10.1609/aaai.v36i4.20309](https://doi.org/10.1609/aaai.v36i4.20309). arXiv: [2112.02962](https://arxiv.org/abs/2112.02962).
- [38] Ahlashkari. „*GitHub - ahlashkari/CICFlowMeter: CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is an Ethernet traffic Bi-flow generator and analyzer for anomaly detection that has been used in many Cybersecurity datasets such as Android Adware-General Malware datas*”. 2022. URL: <https://github.com/ahlashkari/CICFlowMeter> (term. wiz. 2023-09-11).
- [39] Iman Sharafaldin, Arash Habibi Lashkai i Ali A Ghorbani. „*IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*”. 2018. URL: <https://www.unb.ca/cic/datasets/ids-2017.html> (term. wiz. 2023-09-01).
- [40] Microsoft Learn. „*Create compute clusters - Azure Machine Learning | Microsoft Learn*”. 2023. URL: <https://learn.microsoft.com/en-us/azure/machine-learning/how-to-create-attach-compute-cluster?view=azureml-api-2&tabs=python> (term. wiz. 2023-09-11).
- [41] IBM. „*Sposób działania algorytmu SVM - IBM Documentation*”. URL: <https://www.ibm.com/docs/pl/spss-modeler/saas?topic=models-how-svm-works> (term. wiz. 2023-09-13).
- [42] Statsoft. „*SVMIntro3.gif (obraz GIF, 358×131 pikseli)*”. URL: <https://www.statsoft.pl/textbook/graphics/SVMIntro3.gif> (term. wiz. 2023-09-13).
- [43] LightGBM. „*Features — LightGBM 4.0.0 documentation*”. URL: <https://lightgbm.readthedocs.io/en/stable/Features.html> (term. wiz. 2023-09-16).
- [44] Google. „*Lasy decyzyjne | Machine Learning | Google for Developers*”. URL: <https://developers.google.com/machine-learning/decision-forests/intro-to-decision-forests-real?hl=pl> (term. wiz. 2023-09-16).
- [45] Suvres2023. „*GitHub - Suvres/gnb-gp: comparison of GNB with GNB-GA*”. URL: <https://github.com/Suvres/gnb-gp> (term. wiz. 2023-09-15).
- [46] Danet. „*GitHub - WhatAShot/DANet: DANets (a family of neural networks) for tabular data classification/ regression.*” URL: <https://github.com/WhatAShot/DANet> (term. wiz. 2023-09-01).